République Algérienne Démocratique et Populaire

Ministère d'Enseignement Supérieur et de la Recherche Scientifique
Université ABBAS LAGHROUR- Khenchela-
Département MI

Mémoire
Présenté en vue d'obtenir le diplôme de

Master en Informatique (LMD)

Spécialité : Génie logiciel et systèmes distribués (GLSD)

Workflow scientifique pour la composition de services web

Réalisé Par :

- Merkiche Houria
- Abbes Kenza.

Dirigé par :

- ***Dr***.Chouhal Ouahiba

Membres de jury :

- Mm Bardou Dalal (MCB).
- Mr Bakhouch Abedalali (MCA).

Année Universitaire : 2021/2022


بسم الله الرحمن الرحيم

Remerciement

Nous remercions dieu tout puissant pour la santé, les volontés, et sur tout le courage et la patience qu'il nous a donné durant la période de réalisation de ce projet.

Notre témoignage de reconnaissance et gratitude à notre valeureux encadreuse Madame : OUAHIBA CHIOUHAL pour le soutient sincère, ses conseils efficaces et son encouragement pour faire toujours le mieux.

Nous remercions respectivement :

D'avance les membres du jury pour les efforts déployés à l'évaluation de notre modeste.

EN fin nous remercions tous chaleureusement.

Table des matières :

Liste de figures

Liste de tableau

Résumé :

La technologie des services Web est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web. Elle repose essentiellement sur une représentation standard des interfaces et messages au moyen du langage XML. Ils sont utilisés aussi pour fournir une solution à une tâche complexe par regroupement des services web pour créer des services composites. Créer des compositions de services signifie ordonner les invocations aux opérations, router les messages, modifier les paramètres et gérer les exceptions. Donc la composition des services web requiert plusieurs tâches et nécessite un achèvement technique formel. La technique choisie dans ce travail est le workflow scientifique. Ce dernier est modélisés sous la forme d'un diagramme d'activités offre plusieurs interfaces utilisateur pour permettre aux utilisateurs de relier toutes les activités à son flux de travail.

Ainsi, L'objectif principal de notre travail est d'appliquer le workflow pour composer les services web et créer des services composés. La composition des services web par workflow est réalisée par l'utilisation de langage BPEL, l'un des standards les plus couramment utilisés par les organisations de services Web. Nous avons appliqué cette technique pour modéliser le processus de la régulation de la température dans un système de chauffe-eau solaires.

Introduction générale

Les *services Web* sont un nouveau domaine de recherche émergent pour l'informatique orientée services et leur mise en œuvre dans les architectures orientées services. Les services Web sont des applications ou des composants modulaires autonomes fournissant des services. Les services Web peuvent être dynamiquement agrégés, composés et exécutés en tant que *flux de travail* (workflow) de services Web. Ainsi dans ce travail, un workflow scientifique est défini quant à lui, comme un modèle informatique réalisant une fonction complexe décrite par une composition de services Web. Cela nécessite des cadres et des protocoles pourgérer la coordination des résultats de traitement ou des résultats de plusieurs services de manière plus flexible. La norme choisie pour la modélisation de workflow concernant la composition, la coordination des services Web est le langage d'exécution des processus métier pour les services Web (BPEL4WS). Ce choix est motivé par l'intégration de ce langage dans les environnements de développements actuels.

Ainsi, la technique de la composition des services web par workflow est appliquée pour modéliser le processus de la régulation de la température dans un système de chauffe-eau solaires.

Le présent travail comporte trois chapitres :

- Introduction générale

- Chapitre I Service web et Composition

- Chapitre II Workflow scientifique et Bpel

- Chapitre III étude de cas et implémentation

On finira notre travail par une conclusion générale.

Chapitre1 : Service Web et Composition

1. *Introduction :*

L'adoption des services Web constitue une avancée majeure dans le développement des systèmes d'information interopérables. En particulier, la composition de services permet de répondre aux besoins de plus en plus complexes des utilisateurs, par la combinaison de plusieurs services Web au sein d'un même processus métier.

Dans ce chapitre nous allons introduire tous les concepts théoriques concernant le projet. Nous allons commencer par définir brièvement les services web. et Nous présentons plus en détail, par la suite, les services web composite en fournissant la description, le fonctionnement, les différents types.

2. *Les services web :*

2.1 Définition : Il existe plusieurs définitions d'un service web. Nous citons :

Un service Web défini dans [1] est une interface qui décrit un ensemble d'opérations accessibles par le réseau via une messagerie XML standardisée. Un service Web est décrit à l'aide d'une notion XML formelle et standard, appelée sa description de service. Il couvre tous les détails nécessaires pour interagir avec le service, y compris les formats de message (qui détaillent les opérations), le transport protocoles et lieu. L'interface masque les détails d'implémentation du service, lui permettant être utilisé indépendamment de la plate-forme matérielle ou logicielle sur laquelle il est implémenté et également indépendamment du langage de programmation dans lequel il est écrit. Cela permet et encourage les applications basées sur les services Web à être faiblement couplées, orientées composants, implémentations inter-technologies. Les services Web remplissent une tâche spécifique ou un ensemble de tâches. Ils peuvent être utilisé seul ou avec d'autres Web Services pour réaliser une agrégation complexe ou un business transaction.

Un service Web défini dans [2] est un système logiciel conçu pour supporter l'interaction interopérable de machine à machine sur un réseau. Il possède une interface décrite en WSDL (Web Services Description Langage) qui est un format exploitable par la machine. D'autres systèmes interagissent avec les services Web d'une façon prescrite par sa description en utilisant des messages SOAP (Simple Object Access Protocol), typiquement en utilisant http avec une sérialisation XML en même temps que d'autres normes du Web».

2.2 L'intérêt d'un Service Web

Les services Web fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde.

Les services Web sont normalisés car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement. Ils sont donc indépendants des plates-formes. C'est dans ce contexte qu'un intérêt très particulier a été attribué à la conception des services Web puisqu'ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises. Cela s'explique par le fait que les services Web n'imposent pas de modèles de programmation spécifiques. En d'autres termes, les services Web ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes. Cela permet aux vendeurs d'outils de développement d'offrir différentes méthodes et interfaces de programmation au-dessus de n'importe quel langage de programmation, sans être contraints par des standards comme c'est le cas de la plate-forme CORBA qui définit des ponts spécifiques entre le langage de définition IDl et différents langages de programmation. Les services Web représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants [3].

2.3 Les caractéristiques d'un service Web :

La technologie des services Web repose essentiellement sur une représentation standard des données (interfaces, messageries) au moyen du langage XML. Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web.

Un service Web possède les caractéristiques suivantes :

- il est accessible via le réseau ;
- il dispose d'une interface publique (ensemble d'opérations) décrite en XML ;
- ses descriptions (fonctionnalités, comment l'invoquer et où le trouver ?) sont stockées dans un annuaire ;
- il communique en utilisant des messages XML, ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert tels : SMTP, FTP, BEEP...) ;
- l'intégration d'application en implémentant des services Web produit des systèmes faiblement couplés, le demandeur du service ne connaît pas forcément le fournisseur. Ce dernier peut disparaître sans perturber l'application cliente qui trouvera un autre fournisseur en cherchant dans l'annuaire.

2.4 Architecture orienté service Web :

L'architecture orienté services Web repose sur un mécanisme de transport d'une demande de service entre un client et un serveur, tous les deux connectés au réseau.

Dans cette architecture, le client peut être soit un navigateur Web (la demande de service résulte alors directement d'une intervention humaine), soit une application (la demande de service est alors automatisée).

Le bus de requêtes est fondé sur TCP/IP et sur HTTP (mais aussi sur SMTP ou POP), ce qui permet son utilisation sur Internet en vue de l'intégration d'applications au sein du réseau interne de l'entreprise ou de la« publication» d'applications préexistantes sur le Web à destination des entreprises partenaires. Comme http ne transmet que du texte, tous les échanges entre services Web (requêtes et résultats de requêtes) circulent également au format texte, sous forme de documents codés en XML (les requêtes, leurs résultats et les erreurs éventuelles résultant de leur invocation).

2.5 Fonctionnement orienté services Web :

Le fonctionnement dans ce type d'architecture s'articule autour de trois acteurs principaux illustrés par le schéma suivant :

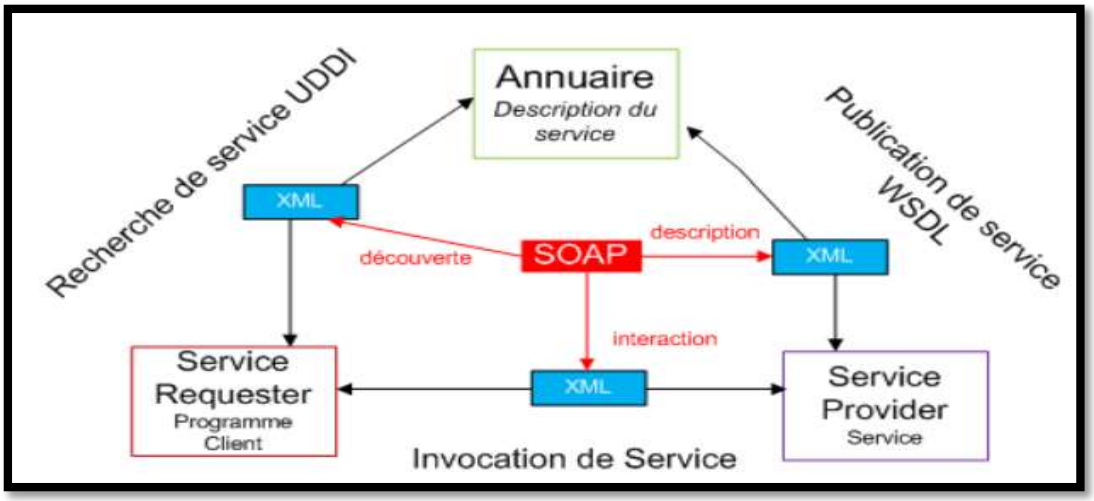


Figure1.1 : fonctionnement orienté service web

Décortiquons ce schéma :

- **Service provider service :** Le fournisseur de service met en application le service Web et le rend disponible sur Internet.

- **Service requester programme client :** C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion réseau et en envoyant une demande en XML (REST, XML-RPC, SOAP).
- **Annuaire service registry :** Le registre de service est un annuaire de services. Le registre fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver.

Ainsi, Les interactions entre ces trois acteurs suivent plusieurs étapes :

- **La publication du service** : le fournisseur diffuse les descriptions de ses services Web dans l'annuaire.
- **La recherche du service :** le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir les descriptions et les URL des services demandés afin de lui permettre de les invoquer.
- **L'invocation du service :** une fois que le client récupère l'URL et la description du service, il les utilise pour l'invoquer auprès du fournisseur de services.

3 Services Web composites :

3.1. Définition de la composition :

La composition est le fait de combiner les fonctionnalités de plusieurs services Web au sein d'un même processus métier pour répondre à une demande complexe qu'un seul service ne pourrait pas satisfaire. Un processus métier est une représentation concrète des tâches à accomplir dans une composition [4].

La réalisation d'une composition nécessite les étapes suivantes :

1. La découverte est le processus de recherche de services Web qui peuvent participer à la composition. Ce processus se fait généralement de manière manuelle par envoi de requêtes aux registres de l'annuaire UDDI [5].
2. L'organisation des interactions entre les services Web dans une composition est définie par deux techniques : la technique d'orchestration (figure 2.1) et la technique de la chorégraphie. Ce travail s'intéresse à l'orchestration qui permet aux différents services d'échanger les messages entre eux. Une composition est associée à une spécification pour gérer les échanges de messages et mettre en place les structures de contrôle nécessaires [6]. Dans la littérature, plusieurs langages de spécification ont été proposés : WSCI, WSFL, XLANG et, plus récemment, YAWL, XPDL, BPMN et WS-BPEL [7] ;

3. L'exécution de la composition est l'étape d'invocation effective des services Web participant à une composition.

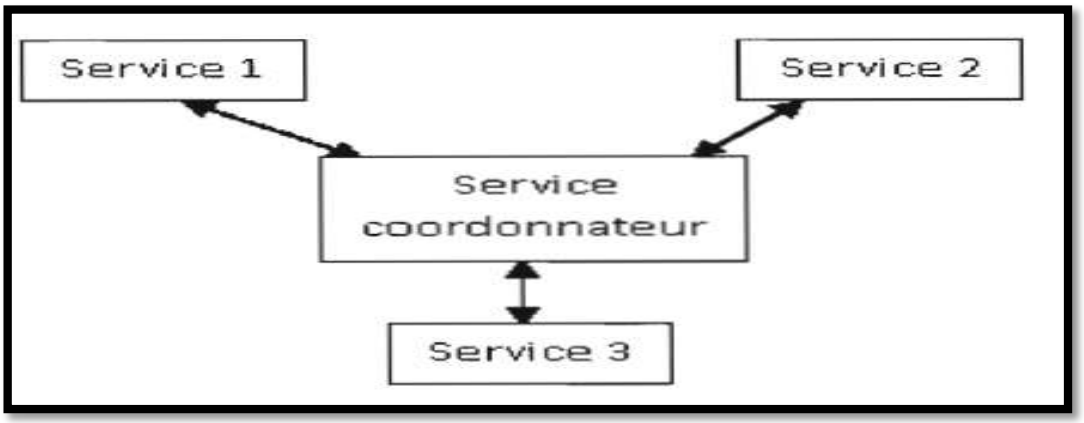


Figure 2.2 : composition par orchestration

3.2. Cycle de vie de la composition :

La composition est un processus relativement complexe. Elle passe par plusieurs étapes. Nous présentons dans la (figure 3.1) un cycle de vie qui montre l'enchainement des étapes d'une composition :

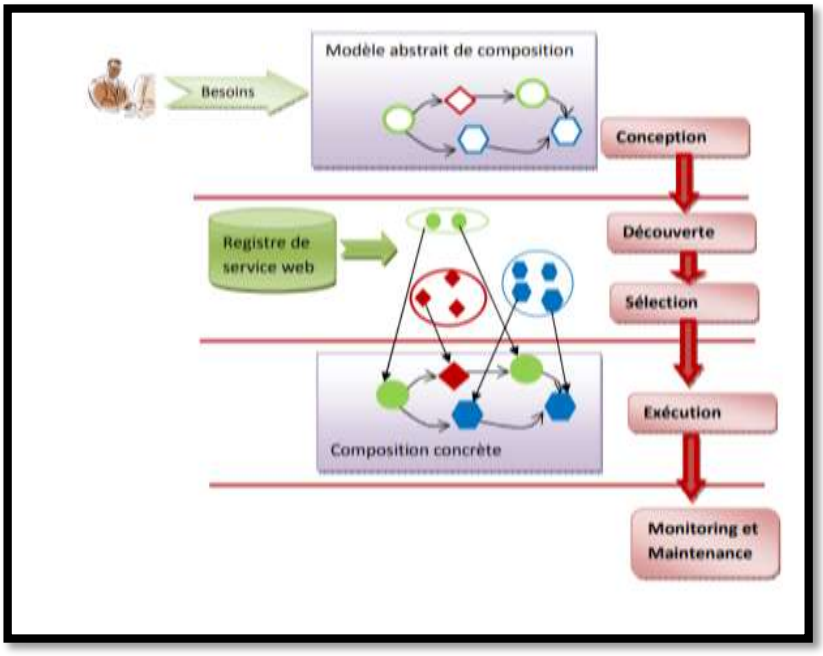


Figure 2.3:L'enchainement des étapes d'une composition

.

a) Etape de la conception :

Après la spécification des besoins fonctionnels, deux modèles doivent être définis :

- **Modèle abstrait :** Il définit la nature des éléments qui seront utilisés dans la composition et le langage utilisé pour définir l'ordre dans lequel les services seront appelés. Parmi les possibilités des modèles abstraits, nous pouvons citer les Workflow, les diagrammes d'activité et les réseaux de Petri.
- **Modèle de données et d'accès de données :** définit comment les données sont spécifiées et comment elles sont échangées entre les composants.

b) Etape de découverte :

La phase de découverte de services permet de trouver à chacune des tâches participantes les services web correspondants. I.e. les services web concrets qui correspondent aux différentes tâches sont localisés. La localisation des services est faite à travers des recherches effectuées dans des registres tels qu'UDDI.

c) Etape de la sélection :

À l'étape présidente, il est très probable que plus d'un service candidat soit trouvé pour chaque tâche. La sélection de services web est l'étape qui vient juste après la découverte. À cette étape, deux modèles doivent être définis :

- **Modèle de sélection de service :** définit si les services sont reliés statiquement ou dynamiquement. Dans le cas de la composition statique, les services sont sélectionnés durant la conception. Dans le cas de la composition dynamique, les services sont déterminés et composés en temps d'exécution.
- **Transactions :** définissent quelles sémantiques de transactions peuvent être associées à la composition, et comment cette association est faite.

Après avoir trouvé les services candidats pour toutes les tâches et lier chaque tâche à son service web choisi, le service composite concret est créé.

d) Etape de l'exécution :

Pendant l'étape d'exécution, une instance du processus est créée en exécutant le service composite. A cette étape on peut définir un modèle pour la manipulation des exceptions qui définit comment les situations exceptionnelles qui produisent l'exécution d'un service composite peuvent être gérées, sans conduire à l'arrêt du service composite.

e) Etape de monitoring et maintenance :

Cette étape permet de contrôler le processus continuellement pour de nouvelles réponses en cas d'un quelconque échec (service indisponible) dans le but de maintenir le bon déroulement de la composition.

3.3. Méthodes et classification des approches de compositions :

La composition de services Web peut être classifiée en fonctions du degré d'automatisation. D'après cette classification on trouve trois catégories de compositions [8]:

Composition manuelle : L'utilisateur génère la composition à la main sans l'aide d'outils dédiés ;

Composition semi-automatique : Dans le processus de composition, l'utilisateur utilise des outils pour aider à la découverte et à la sélection des services les mieux adaptés à son besoin ;

Composition automatique : Le processus de composition est réalisé automatiquement sans l'intervention de l'utilisateur.

Selon les techniques utilisées dans le processus de composition, on peut classifier de la manière suivante les approches pour la composition :

L'approche industrielle : Cette approche est basée sur la description syntaxique de service Web et les technologies liées (WSDL, SOAP, UDDI, etc.) dans la création de processus de composition. WS-BPEL est l'outil le plus utilisé dans ce genre d'approches ;

L'approche sémantique : Une orientation sémantique pour la composition de services Web. Ce genre d'approche utilise les techniques du Web sémantique pour que le processus de composition se réalise automatiquement ;

L'approche formelle : Cette approche utilise des techniques de modélisation et de validation formelle de processus (exemple : les réseaux de Pétri) [9] .

3.4. Langages de définition de la composition par orchestration:

Il existe plusieurs langages de définition pour la composition de services Web. Dans cette section nous allons présenter brièvement quelques-uns d'entre eux.

XLANG :

XLANG est une extension de WSDL créée par Microsoft. Il fournit un modèle pour une orchestration des services et des contrats de collaboration entre ceux-ci.

WS-BPEL (Business Process Execution Language for Web Services):

BPEL est un langage qui se base sur XML [10] . Il a été conçu spécifiquement comme un langage pour la définition des processus métier. Il supporte deux types différents de processus :

1. Les processus exécutables permettent de spécifier les détails du processus métier. Ils peuvent être exécutés au moyen d'un engin d'orchestration [11].

2. Les abstracts business protocois permettent de spécifier l'échange de messages entre partenaires du processus [12].

4. Conclusion :

En conclusion, on peut dire que les services web sont des technologies émergentes et prometteuses pour développer, publier et intégrer des applications Internet. De nombreux services sur le Web ont été créés pour offrir des fonctionnalités plus riches, ce qui a conduit à l'émergence du format dit de service. Ces nouvelles technologies nécessitent le développement d'outils et de techniques pour concevoir des systèmes plus sûrs et plus fiables. Les allocations de service existantes des outils de conception d'entreprise ne fournissent pas de fonctions de vérification formelles. Ainsi, de nombreuses recherches basées sur des formalités diverses ont conduit à la création de nombreux outils dédiés à la vérification des services web. Mais, trop souvent, ces outils sont limités dans leur portée. Cela a conduit à une préférence pour l'utilisation de l'algèbre généralement acceptée comme formalité plus pratique pour la spécification et l'analyse des systèmes synchrones interactifs et distribués.

Dans le chapitre suivant, nous discuterons l'intérêt et les domaines d'application du workflow, sa position dans l'évolution des Systèmes Informations.

Chapitre2 : workflow scientifique et Bpel

1. Introduction :

Un programme informatique (*que nous appelons également service informatique*) correspond à une unité réalisant une fonction atomique. Des programmes informatiques sont par exemple des programmes exécutables, des fonctions de logiciels ou des services Web auxquels nous portons un intérêt particulier. Un processus informatique (*ou workflow informatique*) est défini quant à lui, comme un modèle informatique réalisant une fonction complexe décrite par un processus abstrait. Ainsi, un workflow informatique correspond à une composition de programmes informatiques et particulièrement à une composition de services Web.

L'objectif de ce chapitre est de définir précisément ces notions afin de clarifier l'objet de nos travaux.

2. Définition du workflow :

Le terme workflow a été standardisé par le consortium Workflow Management Coalition (*WFMC*) [13] en 1995. Le standard propose un modèle de référence pour la création, le déploiement et le contrôle d'applications de workflow. La *WfMC* définit le workflow comme suit :

«L'automatisation de tout ou partie d'un processus d'entreprise au cours duquel l'information circule d'une activité à l'autre, c'est-à-dire d'un participant (ou d'un groupe de participants) à l'autre, pour action en fonction d'un ensemble de règles de gestion.»

Cette définition désigne un workflow comme une solution permettant l'automatisation d'un processus métier. Un processus se définit comme un enchaînement coordonné d'un ensemble de tâches aboutissant à un résultat bien déterminé. La coordination spécifie le mode de séquencement des tâches ainsi que les données échangées entre les tâches. En d'autres termes, un processus métier peut être vu comme une application. Cette dernière est construite par composition temporelle de tâches, avec éventuellement des dépendances de données entre ces tâches [14].

Il faut être attentif au fait que le workflow ne contribue pas à l'automatisation des tâches proprement dites, mais à l'automatisation de leurs interdépendances à travers de multiples interactions de coopération et de coordination [15].

3. Concepts fondamentaux du workflow :

Les concepts du workflow ont été résumés par les « *3R* » de Marshak [16] :

3.1. Les Routes : (itinéraire du processus de transformation des informations et des documents) : le routage ou la circulation des documents, des informations sou des tâches est la première grande fonction du workflow.

3.2. Les Règles : (procédures d'action) : la gestion des règles de coordination des activités est la deuxième grande fonction du workflow. Elle est complémentaire à la première dans la mesure où

l'itinéraire d'un processus dépend des règles qui définissent à la fois la nature des informations et leur modalité de transition d'une personne à une autre. Ces règles peuvent être simples ou complexes, mais elles sont indispensables au fonctionnement d'un workflow.

3.3. Les Rôles :(les compétences des différents participants) : la troisième grande fonction du workflow consiste à l'affectation des rôles aux acteurs du workflow.

Un rôle est associé à la réalisation d'une ou plusieurs tâches. Celui-ci peut être affecté à plusieurs acteurs et un acteur peut réaliser plusieurs rôles.

4. Système de gestion de Workflow :

L'automatisation d'un processus métier, ou d'une application, est associée à un système de gestion de workflow pour workflow management system (WfMS), connu également sous le nom de moteur de workflow (en anglais workflow engin). Par abus de langage, le terme workflow est couramment utilisé pour désigner le moteur de workflow.

En fonction de l'enchaînement et la circulation des données qui y sont spécifiées, le moteur a pour responsabilité :

— la planification statique ou dynamique de l'ordonnancement des tâches sur une architecture matérielle,

— l'ordonnancement et le lancement effectifs des tâches et

— l'acheminement des données nécessaires pour leur exécution.

4.1. Définition d'un système de gestion de workflow :

Un moteur de workflow est un système de gestion des tâches à accomplir à partir de l'information fournie en entrée et qui correspond à la composition de ces tâches.

Un WfMS supporte les fonctionnalités des workflow par le biais de deux modules : *modélisation et exécution* où le module de support à la modélisation fournit les primitives nécessaires à la définition des composants. Ces composants sont : activités, entités responsables de l'exécution des activités, données et flux de contrôle entre les activités et enfin, les conditions de début (pré) et de fin (pro) d'exécution des activités.

La WfMC a résumé le rôle d'un moteur de workflow par la définition suivante [17] :

«Un système de gestion de workflow est un système qui complètement définit, gère et exécute les workflow à travers l'exécution de logiciels dont l'ordre d'exécution est dirigé par une représentation informatique de la logique du workflow.»

5. Classification des workflow :

Il existe plusieurs classifications du workflow et celle proposée par la WfMC est la plus adoptée dans la plupart des ouvrages. Elle est utile pour représenter de manière fonctionnelle les différentes applications de workflow. Cette classification est résumée dans la Figure 2.1.

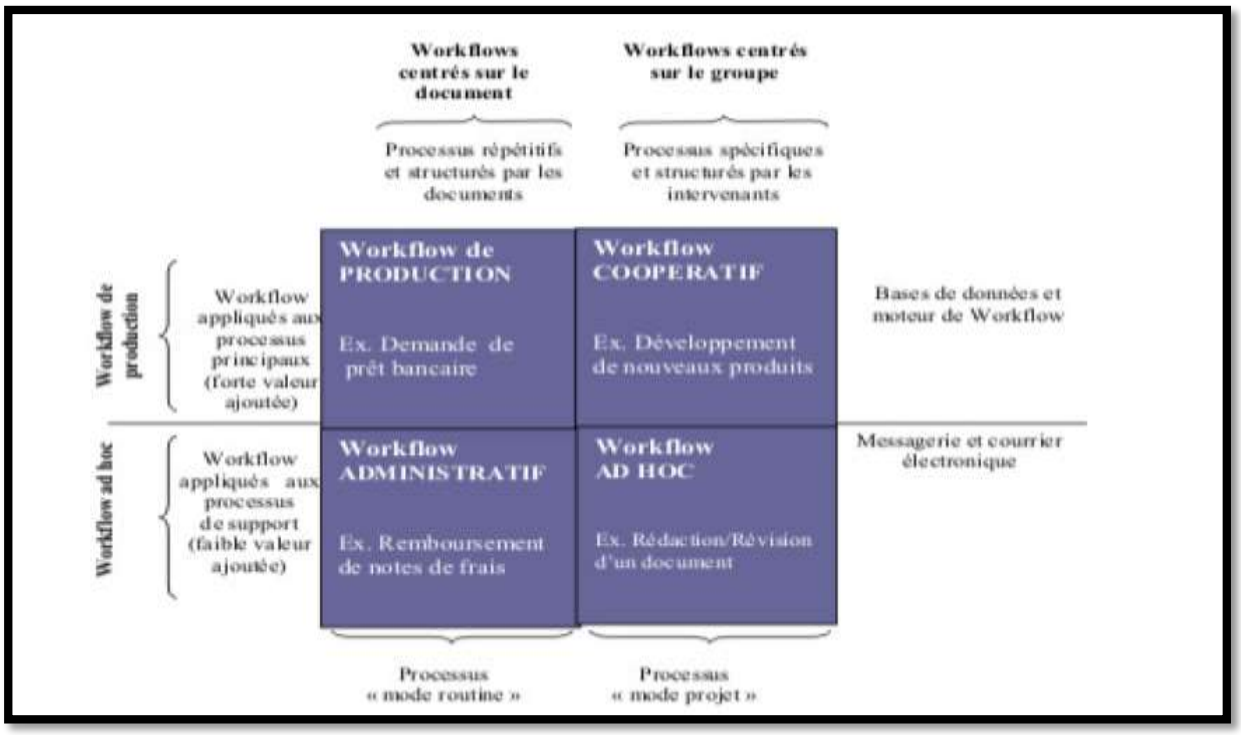


Figure 2.1 : Classification des workflow.

5.1. Workflow de production :

Les processus sont au cœur du métier de l'entreprise et représentent pour elle un niveau de risque élevé. Les tâches effectuées dans le cadre de workflow de production changent peu et elles sont récurrentes. Elles impliquent la participation de plusieurs départements de l'entreprise et l'existence d'une structure créée pour les mettre en place et les contrôler (Exemple de procédures transactionnelles : instruction de demandes de prêts bancaires et traitement des réclamations déposées par les compagnies d'assurance).

5.2. Workflow administratif :

Fondé sur la messagerie et ses extensions, ce type de workflow gère les tâches «administratives» répétitives (approbation des dépenses, demande d'achat, demande de billets pour les voyages, congés, etc.).

5.3. Workflow ad hoc :

Ici, il s'agit de tâches qui sont plutôt associées à des projets qu'à des traitements intensifs. Si les workflow de production gèrent des tâches répétitives, les workflow ad hoc sont soumis à des objectifs dont les étapes et les niveaux d'interaction entre les intervenants sont plus difficiles à

définir en détail et à prévoir (Exemples : Activités liées à un nouveau produit, un marché, l'embauche d'un candidat, etc.).

5.4. Workflow collaboratif :

Le workflow collaboratif est souvent considéré comme du groupware [18]. Il se concentre sur le travail d'équipe en vue d'atteindre des objectifs communs. La taille des groupes peut-être très variable. Elle peut aller du petit comité avec une organisation orientée projet, au grand groupe réparti à travers le monde et ayant des intérêts en commun [19].Les workflow collaboratifs ont pour but de faciliter les communications inter-groupes (Exemple : La gestion des processus plus ou moins formalisés de définition d'un nouveau produit [20]).

Actuellement les workflow sont utilisés pour décrire des expériences scientifiques.

6. Workflow scientifique :

Dans le contexte de la science, les workflow sont utilisés pour décrire des expériences scientifiques. Un workflow scientifique est un flux de tâches principalement des tâches de calcul, qui font partie d'une expérience scientifique. Les workflow scientifiques sont exécutés sur des systèmes distribués vu leur grande demande en capacités de calcul et de stockage [21].

Il existe beaucoup de similarités entre les workflow de type business et les workflow scientifiques. En effet, les workflow scientifiques trouvent leurs racines dans les processus humains et les processus métier. Ils sont vus comme une adaptation, des modèles de type business pour la conception d'applications distribuées, typiquement pour les applications de calcul intensif sur des données massives et pour l'allocation dynamique de ressources d'exécution, en particulier sur les grilles de calcul [22].

Néanmoins, certaines différences séparent ces deux types de workflow. Ces différences sont résumées dans ce qui suit [23]

- Un workflow humain est une séquence de tâches effectuées par une ou plusieurs personnes pour atteindre un certain objectif.
- Les workflow automatiques ou partiellement automatiques se produisent dans les opérations de fabrication d'ordonnancement, la gestion des stocks et la gestion des processus métier. Les workflow de type business focalisent sur l'optimisation de l'efficacité, car ils sont gérés de façon répétée sur un long calendrier et changent rarement.

7. Conception d'un workflow scientifique :

Les systèmes de workflow scientifiques ont conçu beaucoup d'interfaces utilisateur qui permet à un scientifique de relier toutes les activités dans leur propre workflow. En effet, un workflow scientifique est souvent modélisé comme un graphe d'activités. Une activité est un bloc unique de processus qui peut être lié à un autre bloc de traitement si une dépendance de contrôle ou de données existe entre eux.

Dans un graphe, les activités sont faiblement couplées montrant un minimum de communication entre elles.

Selon [24], la conception d'un workflow inclut quatre facteurs clé, à savoir : (a)la structure du workflow, (b) la spécification du workflow, (c) la composition du workflow et (d) les contraintes de qualité de service.

7.1. Structure d'un workflow :

Le workflow est constitué de la connexion de multiples tâches selon leurs interdépendances. La structure d'un workflow aussi appelée pattern [25], indique la relation temporelle entre ces tâches. En général, un workflow peut être représenté par un graphe acyclique dirigé DAG pour (Directed Acyclic Graph) ou un non-DAG (Non-Directed Acyclic Graph) (voir Figure 2.2).

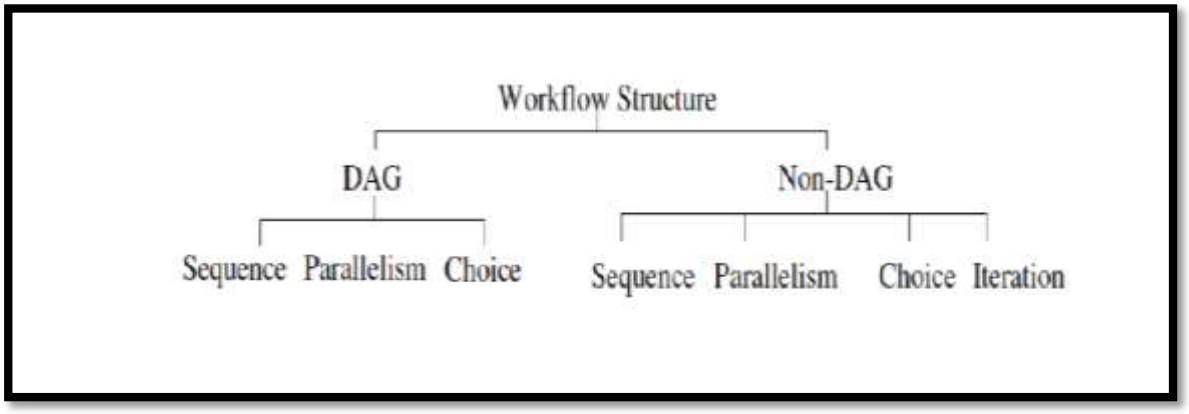


Figure 2.2 : La structure d'un workflow.

Dans un workflow à base de DAG, la structure de tout workflow peut être classée comme une séquence, un parallélisme ou un choix. La séquence est définie comme une série ordonnée de tâches qui commencent chacune à la fin de sa précédente. Le parallélisme représente les tâches qui sont exécutées de façon concurrentielle. Dans le modèle contrôlé par un choix, une tâche est sélectionnée à être exécutée en temps réel quand les conditions qui lui sont associées sont vérifiées.

En plus de tous les patterns contenus dans un DAG, un non-DAG inclut également la structure d'itération dans laquelle, des sections de tâches d'un bloc d'itération sont autorisées à se

répéter. L'itération est aussi un cycle ou une boucle. La structure de l'itération est plus fréquente dans les applications scientifiques.

Ces quatre structures de workflow décrites précédemment, à savoir, la séquence, le parallélisme, la condition et l'itération peuvent être utilisées pour construire des workflow complexes. De plus, des sous workflow peuvent également se baser sur les mêmes structures dans la construction de blocs pour la modélisation d'workflow à large échelle.

7.2. Spécification d'un workflow :

La spécification d'un workflow aussi appelée modélisation définit un workflow contenant la définition de ses tâches et de sa structure. Souvent les concepteurs de workflow scientifiques distinguent deux types de modèles : le modèle abstrait et le modèle concret. Ils représentent respectivement les tâches constituant un workflow avec leurs interdépendances et l'ensemble des ressources physiques impliquées dans l'exécution du workflow [26].

7.3. Composition d'un workflow :

Les systèmes de composition de workflow sont désignés pour permettre aux utilisateurs d'assembler des composants en un workflow. Ils doivent fournir une vue de haut niveau de la construction des applications des utilisateurs et cacher la complexité des systèmes. Dans [27], les auteurs présentent un état de l'art des classes principales de tous les outils et langages de la composition de workflow. Cet état de l'art est montré dans la (Figure 2.3) où les auteurs donnent une terminologie dédiée aux systèmes de composition de workflow.

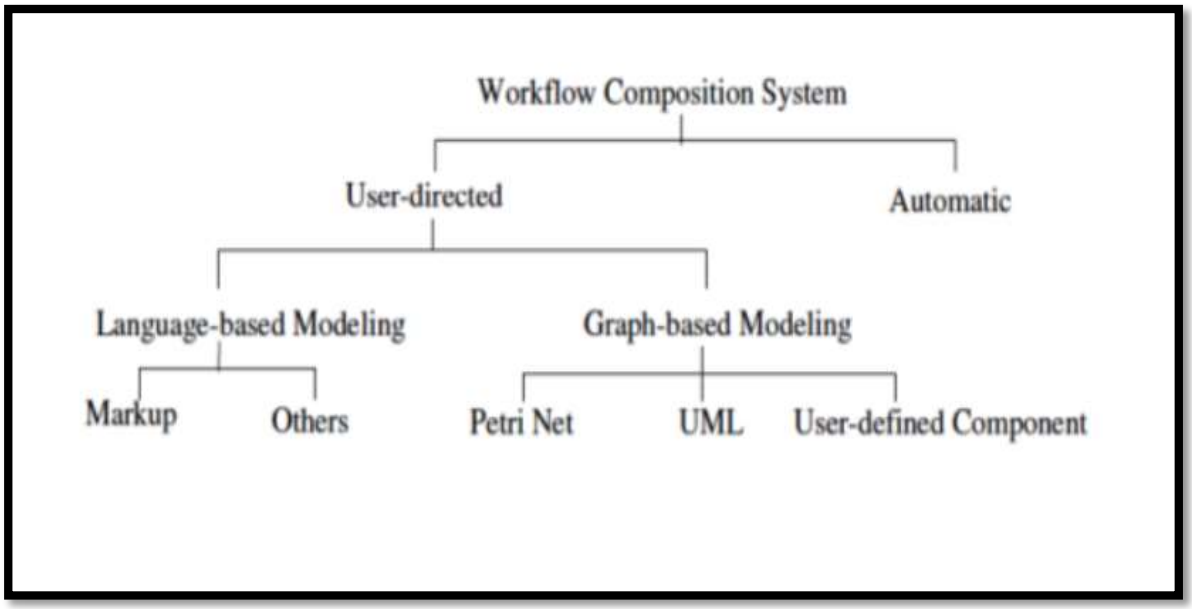


Figure 2.3 : La composition d'un workflow.

Les systèmes orientés utilisateurs permettent à ces derniers de modifier directement leurs workflow tandis que les systèmes de composition automatique génèrent les workflow automatiquement. En outre, les avantages d'une composition automatique sont :

(1) la sélection des composants du workflow est effectuée de façon automatique. Ceci est utile dans le cas où des centaines de milliers de composants sont concernés par cette sélection ;

(2) des workflow alternatifs peuvent être modélisés automatiquement. Dans le cas a où un composant n'est pas disponible ou tombe en panne, un workflow alternatif peut être adopté ;

(3) les workflow conçus automatiquement peuvent être optimisés afin de pouvoir être exécutés dès que les ressources et les objets d'entrée sont disponibles.

7.3.1 Modélisation à base de langage :

Lors d'une modélisation à base de langage, les utilisateurs peuvent exprimer leur workflow en utilisant des langages à balises tel que : Extensible Mark up Language(XML)[28],GridAnt[29],WSFL[30],BPEL4WS[31] , W3C-XML-Pipeline[32],, GridbusWorkflow[33],ou d'autres formats comme le Condor DAGman[34].

7.3.2. Modélisation à base de graphe :

La modélisation à base de graphe permet une définition graphique d'un workflow. Elle permet aux utilisateurs de composer et de mettre à jour le workflow par un simple clic sur les composants qui les intéressent. Les approches les plus répandues sont celles basées sur les réseaux de Petri [35] et UML (Unified Modeling Language) [36].

7.4 Qualité de Service :

Les utilisateurs peuvent sélectionner les ressources appropriées et les utiliser pour leurs applications de workflow [37]. Les ressources peuvent fournir la même fonctionnalité, mais optimisent les différentes mesures de qualité de service QoS (pour Quality of Service) [38].

Souvent les modèles de QoS comportent cinq dimensions [39] : le temps, le coût, la fidélité, la fiabilité et la sécurité.

- Le temps est une mesure de base de la performance. Pour les systèmes de workflow, il se réfère au temps total nécessaire pour achever l'exécution d'un workflow.
- Le coût est associé à l'exécution des workflow et comprend le coût de gestion des systèmes de workflow et la charge d'utilisation des ressources de la grille pour des tâches de traitement de workflow.
- La fidélité se réfère à la mesure liée à la qualité du rendement de l'exécution du workflow.
- La fiabilité est liée au nombre de défaillances pour l'exécution des workflow.

- La sécurité fait référence à la confidentialité de l'exécution de tâches du workflow et la fiabilité des ressources.

8. De la composition de workflow à la composition de service :

Les services composés sont définis récursivement comme étant une agrégation de services élémentaires et composés. En composant des services web, le logique métier du client est implémenté par plusieurs sévices. ceci est analogue aux WfMS (Workflow Management System) [40] où la logique applicative est réalisée en composant des applications autonomes [41]. Ceci permet la définition d'applications de plus en plus complexes en agrégeant progressivement des composants de niveau élevé d'abstraction [42].

Trois éléments principaux d'un système de composition de service web peuvent être identifiés [43] .Ces éléments sont : un modèle de composition et un langage pour spécifier les services impliqués dans la composition, un environnement de développement muni d'une interface graphique et d'un environnement d'exécution pour accomplir la logique métier. De plus, un middleware de composition de service exige que les fonctionnalités, les interfaces et les protocoles que les services web supportés soient décrits avec précision.

Par conséquent, les composants sont spécifiques au système et aux vendeurs et nécessitent un effort de développement supplémentaire.

Le langage de modélisation utilisée pour exprimer le workflow de composition des services web est le langage Bpel « Business Process Execution Language», connu par WS-BPEL.

8.1. LangageWS-Bpel

BPEL supporte la composition via sa capacité à formuler les deux types de processus décrits ci-dessous [44] :

8.1.1.Les processus métier exécutables : Spécifient les détails d'interactions entre les services qu'ils composent ; spécifient les algorithmes exacts contrôlant les activités des compositions à définir, sans oublier de spécifier les messages entrants et sortants échangés par ces activités. La définition de tels processus suit le paradigme de l'orchestration et pourrait être exécutée par un moteur (un engin) BPEL. Dans la majorité des cas, c'est ce type de processus qui est spécifié par le BPEL.

8.1.2.Les processus métier abstraits : Ne spécifient que les échanges publics entre les composants participants (que ce soit des processus ou de simples services), et leur manque de détails d'interaction les résument à des processus non-exécutables. La définition de tels processus suit le paradigme de la chorégraphie. Le cas d'utilisation le plus commun de ce type de processus

correspond à leur usage comme gabarits (templates) aidant à la définition de futurs processus exécutables.

8.2.La spécification WS-BPEL :

Tout processus BPEL est décrit par un document XML ayant une structure de base qui est typiquement semblable à celle présentée ci-dessous. La racine d'un tel document correspond toujours à l'élément <***process***>. Cet élément pourrait disposer de plusieurs attributs. Parmi ces attributs, on cite l'attribut ***abstractProcess*** qui permet de spécifier si le processus est défini comme processus abstrait ou exécutable**.**

```
<process [attributs]>
    <partnerLinks>. . .</partnerLinks>
    <variables>. . .</variables>
    <correlationSets>. . .</correlationSets>
    <faultHandlers>. . .</faultHandlers>
    <eventHandlers>. . .</eventHandlers >
    { Activity }
</process>
```

Mis à part ses attributs, la racine <***process***> dispose aussi d'un ensemble d'éléments enfants (e.g. <***partnerLinks***> et <***variables***» qui sont suivis du jeton ***Activity***. Ces éléments sont optionnels et regroupent des constructeurs qui constituent la partie déclaration du proces-sus. Quant au jeton ***Activity***, il marque la présence des constructeurs qui constituent la partie traitement (***Workflow***) de ce dernier. Ainsi, dans tout processus, on distingue deux catégories de constructeurs : les constructeurs dédiés à sa partie déclaration, et ceux permettant de définir sa partie traitement.

8.2.1. Les constructeurs de déclaration :

Ces constructeurs instaurent une déclaration globale s'ils sont directement liés au proces-sus. En revanche, si de tels constructeurs sont liés à une des unités logiques composant ce dernier (à un <scope>), dans ce cas on dit que ces constructeurs instaurent une déclaration locale. Dans les deux cas, ces constructeurs se définissent comme suit :

Les partnerLinks : Ils sont la modélisation des services avec lesquels le processus métier interagit. Leurs déclarations spécifient la forme statique des relations qu'aura le processus avec ses partenaires (i.e., d'autres processus ou services). Chaque <partnerLink> est typé par un <partnerLinkType>. Ce dernier détermine le rapport conversationnel entre deux services en spécifiant le rôle que joue chacun des deux dans une conversation.

```
<partnerLinks>
    <partnerLink name="ncname"
          partnerLinkType="qname" />+
</partnerLinks>
```

Les variables : Elles permettent de garder trace des données représentant l'état interne du processus. Dans la majorité des cas, ces données correspondent aux messages échangés entre le processus ct ses partenaires. Une variable peut être déclarée comme un type de message WSDL (i.e., un messageType), un type XML Schéma ou un élément XML schéma.

```
<variables>
    <variable name="ncname" messageType="qname"?
          type=''qname''? element=''qname''?/>+
</variables>
```

Les correlationSets : Ils permettent à ce que les messages envoyés soient livrés non pas juste au bon port de destination, mais aussi à la bonne instance de destination du processus considéré.

Pour ce faire, chaque <***correlationSet***> définit un groupe de ***propriétés*** permettant d'identifier, d'une façon unique, chaque instance d'un même processus.

```
<correlationSets>
    <correlationSet name="ncname" properties="qname-list"/>+
</correlationSets>
```

Les faultHandlers : Ils permettent de traiter les erreurs qui seraient déclenchées au cours de l'exécution du processus. Généralement, ils sont conçus dans l'optique d'annuler le travail qui n'a pas pu être achevé dans un processus où une erreur a eu lieu.

Les eventHandlers : Ils permettent au processus, ou à une des unités logiques le composant (i.e., un <scope», de pouvoir traiter les événements ***normaux*** qui seraient déclenchés au cours de son exécution. Ces événements sont de deux types : ceux qui ont lieu suite à la réception d'un message, et ceux qui se manifestent suite au déclenchement d'une alarme.

8.2.2. Les constructeurs de définition du traitement :

Tout processus BPEL dispose d'une seule activité principale. Cette activité pourrait correspondre à l'une des activités structurées ou de base qui sont offertes par le langage BPEL.

Ci-dessous, nous exposons l'ensemble de ces activités.

a. Les activités de base :

Parmi les activités de base les plus importantes, on trouve celles qui sont liées à la réception et à l'envoi des messages, respectivement, de la part et à destination des partenaires. Ces activités sont :

L'activité <*receive*> : Permet aux partenaires de faire appel à une des opérations exposées par le processus avec lequel ils interagissent. Elle bloque j'exécution séquentielle jusqu'à recevoir un message dont les propriétés concordent avec les valeurs des attributs ***portType*** et ***operation***. Une fois reçu, ce message est sauvegardé par la variable qu'elle spécifie. Il faut aussi noter que cette activité joue un rôle clé dans le cycle de vie d'un processus puisqu'elle permet son instanciation si l'attribut ***createInstance*** est à *yes.*

```
<receive partnerLink="ncname" portType="qname"
    operation="ncname" variable="ncname"? createInstance="yes|no"?
</receive>
```

L'activité <*reply*> : Est utilisée pour l'envoi d'un message en réponse à un premier message ayant été reçu à travers une activité <***receive***> précédemment exécutée sur le même ***partnerLink***, ***portType*** et ***operation***. Le résultat d'un <***reply***> peut prendre deux formes : la première forme correspond au cas où le <***reply***> résulte en une réponse normale qui sera portée par la variable qu'il spécifie, tandis que la deuxième correspond au cas où ce dernier résulte en une erreur dont le nom sera spécifié par l'attribut ***faultName***.

```
<reply partnerLink="ncname" portType="qname"
    operation="ncname" variable="ncname"? faultName="qname"?
</reply>
```

L'activité <*invoke*> : Permet l'invocation d'une opération exposée par un service Web. Cette invocation peut être synchrone (sous forme d'une requête/réponse) ou asynchrone (sous forme d'une requête à sens unique).La déclaration d'une invocation synchrone requiert la présence et d'une ***inputVariable*** spécifiant le message à envoyer et d'une ***outputVariable*** pour la sauvegarde du message à recevoir. Contrairement à cela, la déclaration d'une invocation asynchrone n'exige la présence que d'une ***inputVariable***. Une invocation synchrone peut aussi résulter en un message reportant une erreur.

```
<invoke partnerLink="ncname" portType="qname"
    operation="ncname" inputVariable="ncname"? outputVariable="ncname"?
</invoke>
```

En plus des trois activités décrites ci-dessus, le langage BPEL offre d'autres activités de base que nous présentons comme suit :

•**L'activité <assign> :** Permet de copier des données d'une variables à une autre, de même que de construire de nouvelles données à base d'expressions, e.g. expressions XPath.

•**L'activité <throw> :** Permet de signaler une erreur interne d'une façon explicite.

•**L'activité <wait> :** Indique soit une ***durée*** soit une ***date limite*** pendant/avant laquelle l'exécution du processus est bloquée.

•**L'activité <empty> :** Indique au processus de ne rien faire.

•**L'activité <exit> :** Permet de mettre fin à l'exécution d'une instance d'un processus, d'une façon immédiate.

•**L'activité <rethrow> :** Permet de lever (à une autre portée) une erreur ayant été initialement capturée par le <***faultHandler***> englobant.

b. Les activités structurées :

Les activités structurées dotent le langage BPEL de cette capacité à exprimer des compositions de services selon une approche structurée ; approche héritée du langage XLANG. Ci-dessous, nous énumérons l'ensemble de ces activités :

• **L'activité <*sequence*>.** Contient (une à) plusieurs activités, structurées ou de base, qui devront être exécutées en séquence, suivant l'ordre selon lequel elles sont listées.

• **L'activité <*if*>.** Permet de sélectionner, à partir d'une liste ordonnée de choix, une et une seule activité à exécuter.

• **L'activité <*pick*>**. Attend l'occurrence d'un événement parmi plusieurs, pour n'activer qu'une seule branche parmi plusieurs, et par suite n'exécuter qu'une activité parmi plusieurs.

• **L'activité <*while*>.** Permet une exécution répétitive de l'activité qu'elle enveloppe, et cela tant que la condition qu'elle définit est maintenue à *vrai*.

• **L'activité <*repeatUntil*>**. Permet une exécution répétitive de l'activité qu'elle enveloppe, ct cela tant que la condition qu'elle définit est maintenue à *faux*.

•**L'activité <*forEach*>.** Permet d'exécuter, d'une façon parallèle ou *séquentielle,* un certain nombre d'instances de l'activité <scope> qu'elle enveloppe.

• **L'activité <*fiow*>.** Permet à ce que toutes les activités qui y sont directement enchâssées soient concurremment exécutées. À travers des liens de contrôle (i.e., des <***links***>) qu'elle déclare, elle arrive à définir des dépendances de contrôle entre les activités qu'elle enchâsse.

• **L'activité <*scope*>.** Définit un contexte d'exécution complet pour l'activité qu'elle enveloppe. Ce contexte a une structure similaire à celle d'un processus vu dans sa globalité. Un<***scope***> pourrait aussi définir des constructeurs <***compensationHandlers***> et <***terminationHandlers***>. Les premiers permettent de défaire le travail exécuté par le <***scope***> auxquels ils sont attachés, alors que les deuxièmes permettent à ce dernier de contrôler, à un certain niveau, la sémantique de sa terminaison forcée.

9. Conclusion :

Dans ce chapitre, nous avons présenté la technique de workflow (flux de travail) : définitions, concepts, structures de flux de travail et discussion sur l'idée d'intégration flux de travail dans les systèmes distribués orientés services.

Ce chapitre détaille un état de l'art des principaux outils et langages de la modélisation d'un workflow plus particulièrement le langage de modélisation utilisé pour exprimer les workflow de configuration des services web est le Bpel, connu sous l'acronyme WS-BPEL.

Le chapitre suivant a pour objectif de présenter l'application de workflow pour composer les services web et modéliser un processus dans un système réel sous forme d'un service web composite.

Chapitre 3 : étude de cas et implémentation

1. Introduction :

Ce chapitre présente notre cas d'étude qui est le système chauffe-eau solaire. Après l'étude de ce système nous avons constaté que le processus de régulation de température est la plus importante. Nous avons décomposé ce processus en services web élémentaires autonomes ensuit nous avons implémenté et testé ces services. En utilisant le langage BPEL, nous avons modélisé ce processus sous forme d'un workflow qui utilise et appel les services web développés. Le résultat de cette composition est un service web composite donc nous avons testé ce service composite et expliqué son fichier WSDL. Ce chapitre sera également consacré aux aspects de mise en œuvre. Ci-dessous, nous décrivons la plate-forme et l'environnement de programmation utilisés. Nous terminons ce chapitre par une description de notre application web qui permet de tester les services développés.

2. *Étude De cas : Chauffe-eau solaire*

Le chauffe-eau solaire, représenté par la ***Figure3.1***, est l'un des systèmes solaires thermiques qui peuvent être mis en application avec peu de moyens permettant d'obtenir des performances importantes pour la production de l'eau chaude sanitaire pour satisfaire le faible niveau de température demandé d'ordre 45°C à 60°C.

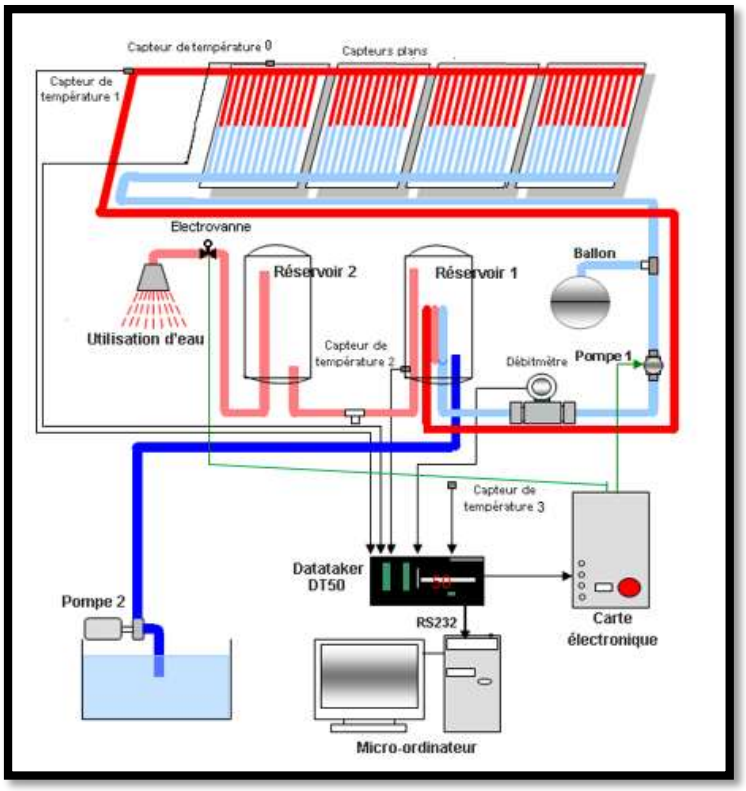


Figure 3.1 : Schéma synoptique du système chauffe-eau solaire**.**

Les rayonnements solaires échauffent le fluide caloporteur circulant dans l'absorbeur du capteur solaire placé sur le toit de l'unité de recherche. Un fluide caloporteur circule dans la

conduite du circuit fermé et conduit la chaleur captée vers l'unité de stockage (réservoir 1) à l'aide d'une pompe de circulation, cette dernière est alimentée par un circuit électrique commandée par un ordinateur. À l'aide de capteurs, les températures au niveau du capteur plan et au niveau du premier réservoir sont mesurées. La pompe est activée dès que la différence entre ces deux températures aura atteint un ΔT donnée et s'arrête dès que cette valeur diminue. Tant que la différence entre les deux températures reste supérieure à la valeur ΔT pré fixée, l'évacuation de l'eau continuera et l'échange de chaleur se fera jusqu'à ce que l'eau dans le réservoir ne soit plus froide, l'eau froide est toujours disponible au niveau du premier réservoir, l'eau chaude récupérée dans le second réservoir est prête pour l'utilisation, lorsque la température atteint celle demandée par l'utilisateur, une électrovanne s'ouvre automatiquement.

2.1 Principe de fonctionnement du chauffe-eau solaire :

Les capteurs plans, l'élément essentiel dans cette installation, sont des plaques qui transforment l'énergie du rayonnement solaire en énergie thermique. Chaque ensemble de capteur est constitué principalement d'un absorbeur. Le principe de fonctionnement d'un capteur est basé sur l'absorption de la chaleur d'une part et sur l'effet de serre d'autre part. L'énergie captée au niveau des capteurs plans est transportée par le fluide caloporteur vers l'échangeur de chaleur. Ce dernier transmet l'énergie captée par les capteurs plans à l'eau à chauffer. L'échangeur utilisé dans ce type de chauffe-eau solaire est un échangeur à serpentin immergé dans le premier réservoir.

Par ailleurs, la régulation mis en place dans cette installation a pour but d'optimiser le fonctionnement du système. Son action est basée sur la comparaison de la température sortante des capteurs plans avec celle correspondante au niveau du premier réservoir. La pompe démarre le moment où l'écart entre ces deux températures est supérieur à 10°C et s'arrête lorsque ΔT est inférieur à cette valeur. L'évacuation de l'eau chaude s'exécute en ouvrant une électrovanne dès que la température réalisée atteint celle demandée par l'utilisateur. Elle est de 40 °C dans le cas de ce chauffe-eau solaire. Le fonctionnement d'écrit est représenté par le diagramme des cas d'utilisation suivant :

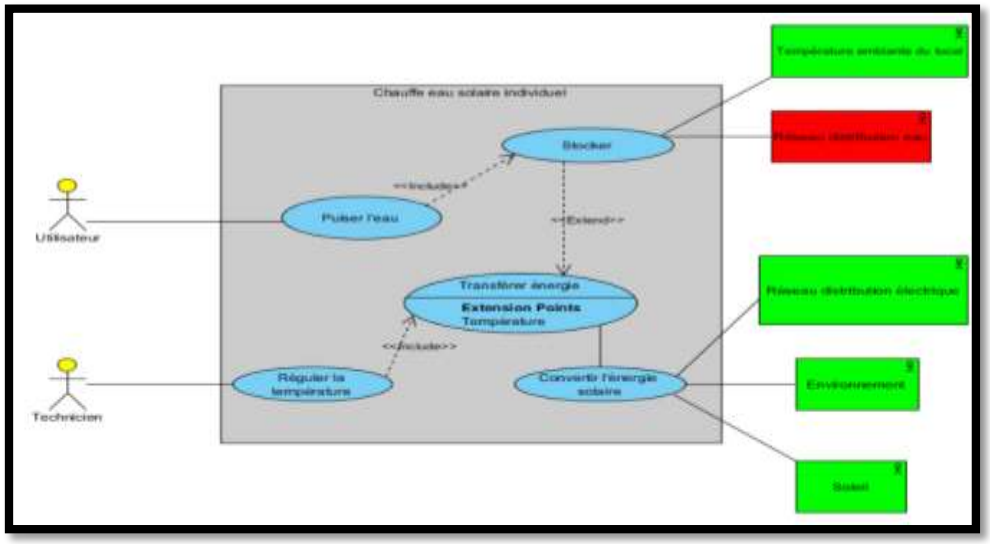


Figure3.2 : Diagramme des cas d'utilisation de chauffe-eau solaire**.**

3. Modélisation de system chauffe-eau solaire

3.1. Présentation d'UML :

UML (Unified Modeling Language) est un langage formel et normalisé en termes de modélisation objet. Son indépendance par rapport aux langages de programmation, aux domaines de l'application et aux processus, son caractère polyvalent et sa souplesse ont fait lui un langage universel. En plus UML est essentiellement un support de communication, qui facilite la représentation et la compréhension de solution objet. Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation des solutions. L'aspect de sa notation, limite l'ambigüité et les incompréhensions.

UML est utilisé pour modéliser un diagramme de cas d'utilisation qui identifie les utilisateurs du système (acteurs) et leurs interactions avec le système, un diagramme d'activités qui résume sous forme d'un organigramme les activités principale implémentés par notre travail et le diagramme de séquence qui permet de représenter des collaborations en objets selon un point de vue temporel, on y met l'accent sur la chronologie (envois de messages).

3.1.1. Diagramme Cas d'utilisation :

Le bon fonctionnement de système chauffe-eau solaire est assuré par plusieurs fonctions (voir figure 3.2). Dans ce travail on s'intéresse uniquement à l'implémentation de la fonction « réguler température » réalisé par un technicien. Dons notre l'implémentation on distingue les cas d'utilisation suivants :

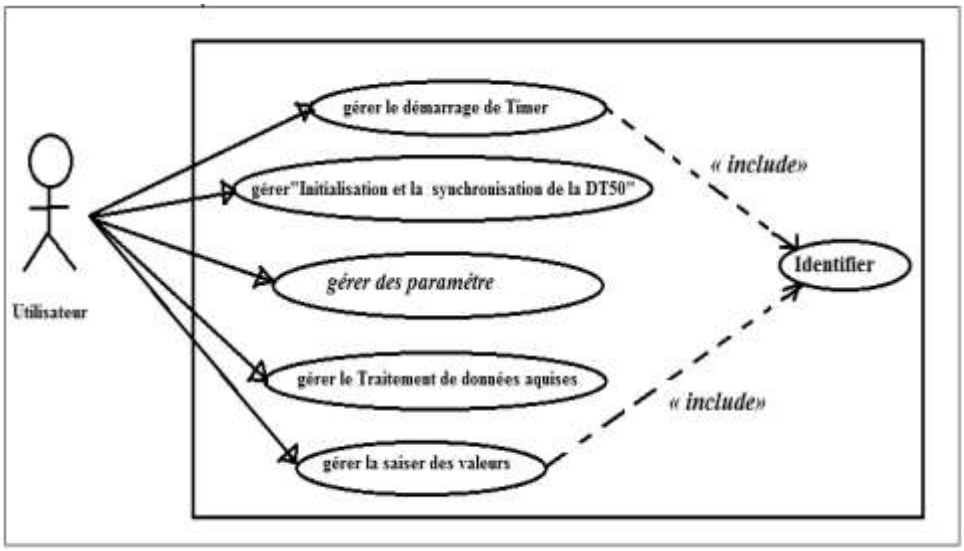


Figure3.3 : Diagramme des cas d'utilisation.

3.1.2. Diagramme d'activités :

La figure3.3 résume sous forme d'un organigramme les activités principale de processus « réguler la température » implémentés par notre travail

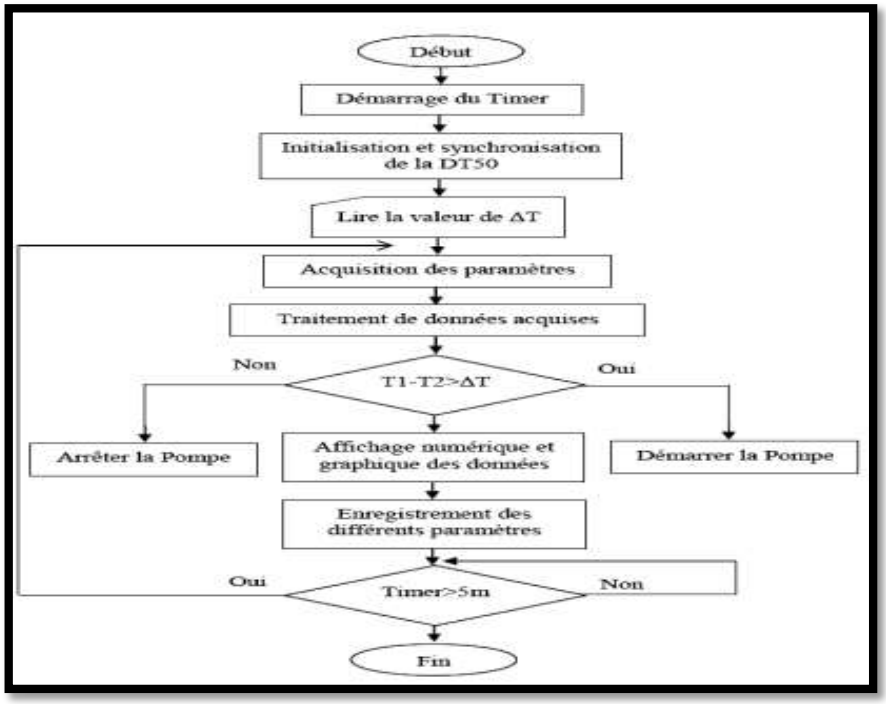


Figure3.4 : activités de processus « réguler la température »

Ce processus est modélisé par les services web élémentaires suivants :

Description des enchaînements	
Nom des services :	Activité :
SW1 : Timer	le processus commencer le démarré de timerSW1 le Timer a été démarré. SW1 le Timer a été démarré.
SW2 :DT50	le processus envoyé initialisation et synchronisation de DT50. SW2 DataTaker50 a été synchrone et initialisé
SW3 :ΔT	Le processus a Demande de lecture de valeurs SW3 les valeurs a été lire
SW4 : Acquis	SW1 envoyé les paramètres SW1 envoyé DT50 SW1 envoyé la valeur de ΔT SW4 Acquise
SW5 : Traitement	SW4 Traité les données SW5 les données a été traité
SW6 : Comparaison SW7 : réponse	SW6 condition a été réalisée SW7 démarrer la pompe
SW8 : Ne répond pas	SW6 condition n'est pas réalisée SW8 Arrêter la pompe
SW9 : Affichage	SW6 Afficher les données numérique et graphique SW6 Enregistrer des différents paramètres SW9 la condition a été réalisée
SW10 : Test	SW9 : Test de condition SW10 a SW3 la comparaison est réalisée SW10 : la comparaison n'est pas réalisée
SW11 : Stop	SW10 : fin de traitement SW11 : le processus a été arrêté

Tableau1 :Description des enchaînements

3.1.3. Diagramme de séquence : Le diagramme de séquence suivant explicite le message échangé par les services web :

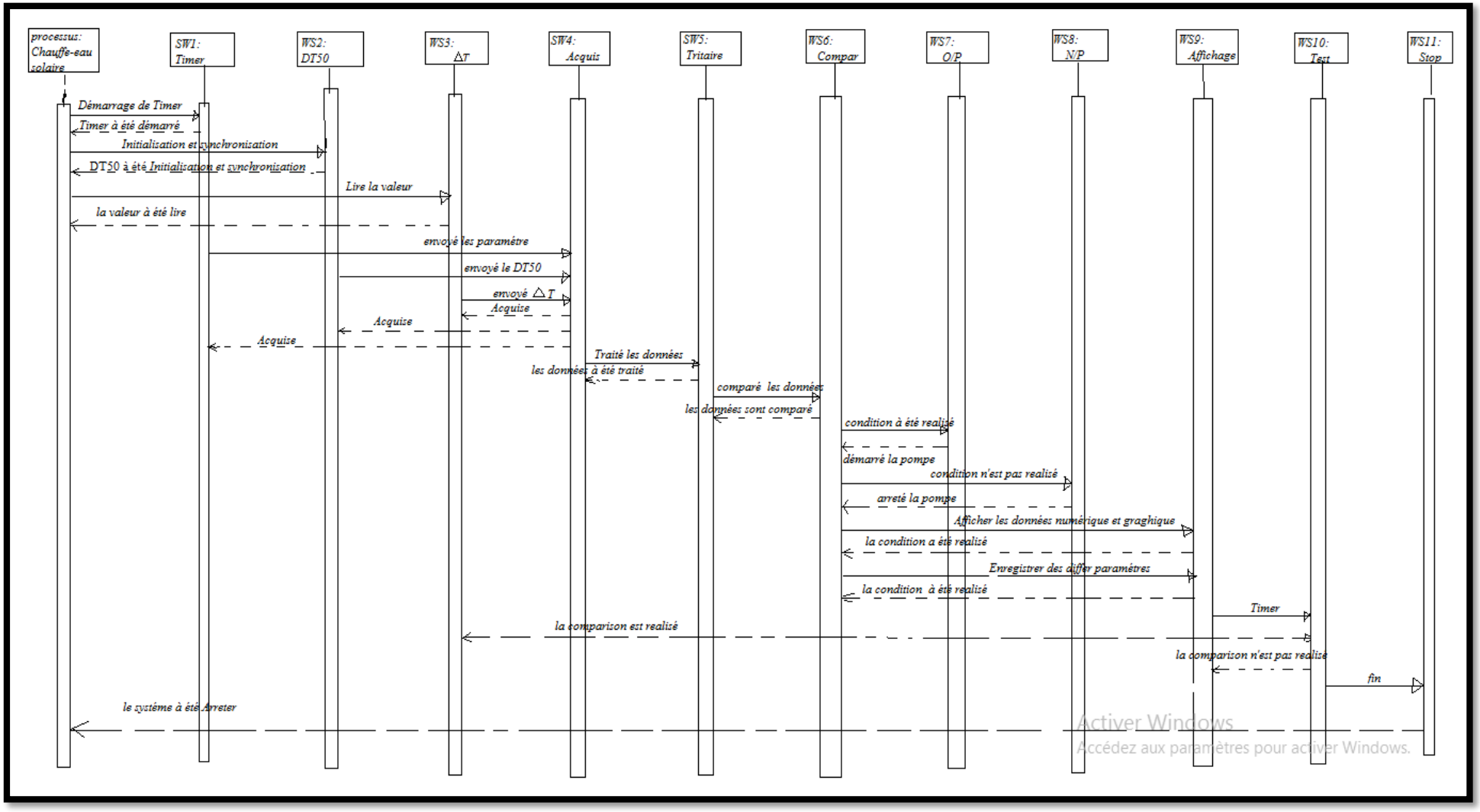


Figure 3.5 : Diagramme de séquence.

4. Configuration logiciel et installation :

- ✓ **JDK 1.6.**

http://www.oracle.com/

- ✓ **NetBeans IDE 6.0.1.**

https://netbeans-ide.informer.com

- ✓ **Serveur Glassfich :**

Intégré par défaut dans cette version de NetBeans.

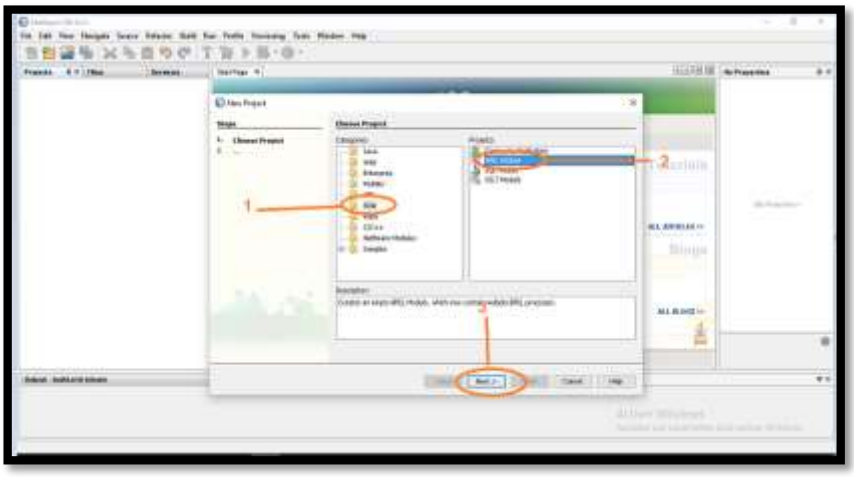
5. Création d'un projet de module BPEL :

La procédure typique à suivre lors de la construction d'un processus BPEL est :

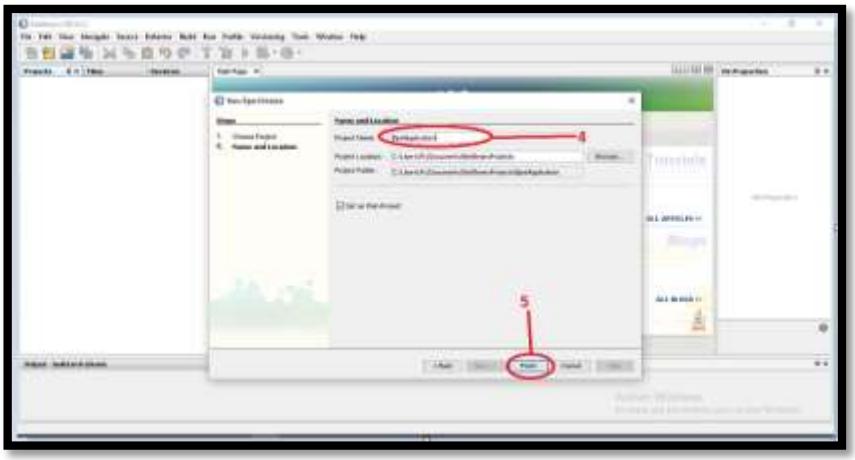
5.1 **Création d'un nouveau projet de module BPEL :**

Pour créer un nouveau processus BPEL:

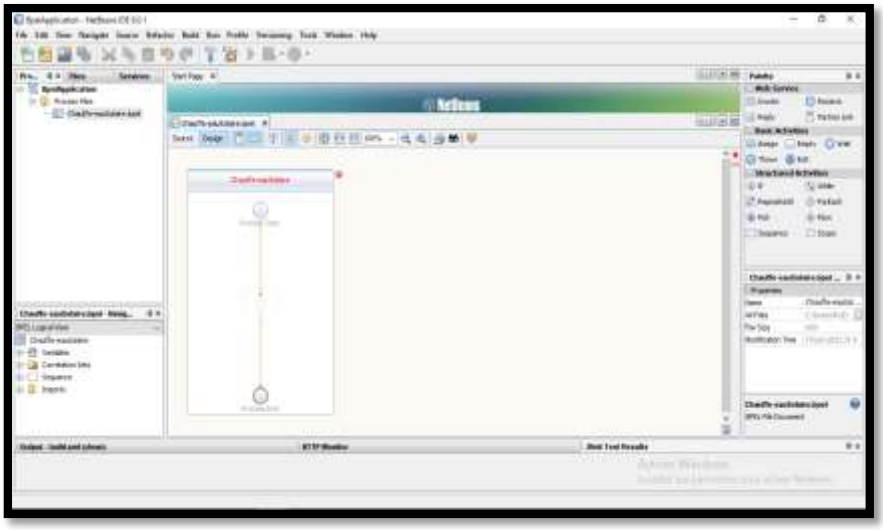
- Aller vers ***New Project***,choisir la catégorie ***SOABPEL ModuleNext***comme la figure en bas:



- Choisir ***BpelApplication***comme nom de ***projet***:



- ***Chauffe-eau Solaire*** comme nome de process, Vous obtiendrez le résultat suivant :

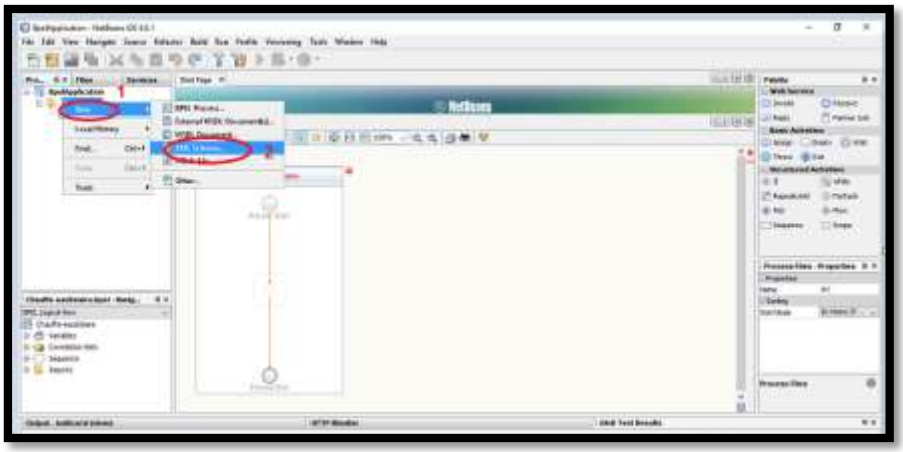


5.2 **Création de fichier XSD (XML schéma Définition) :**

Pour une bonne pratique, nous avons créerun fichier ***XSD*(XML Schéma Définition)** dans lequel nous définissons les types de messages qui seront échangés dans ce processus ***BPEL***.

-Pour créer un nouveau fichier ***XSD***:

- ✓ Faire un clic droit sur le répertoire ***Process Files*** *de votre application* ***BPEL****, et choisir* ***New→XML Schema****.*
- ✓ Appeler ce schéma ***Bpelxsd.***



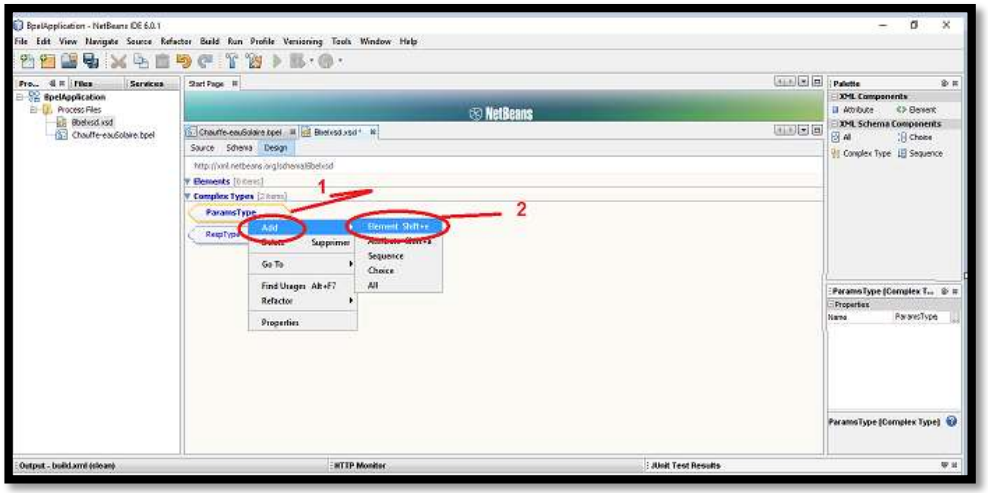
- ✓ Dans la fenêtre qui apparaît, choisir l'onglet ***Design***pourunereprésentationgraphique de votre schéma.comme en bas :

Pour créer des ***TypesoComplexes***, faites glisser ***ComplexType***à partir de la palette à droite de fenêtre principale de***Netbeans***vers le champ ***ComplexTypes.***

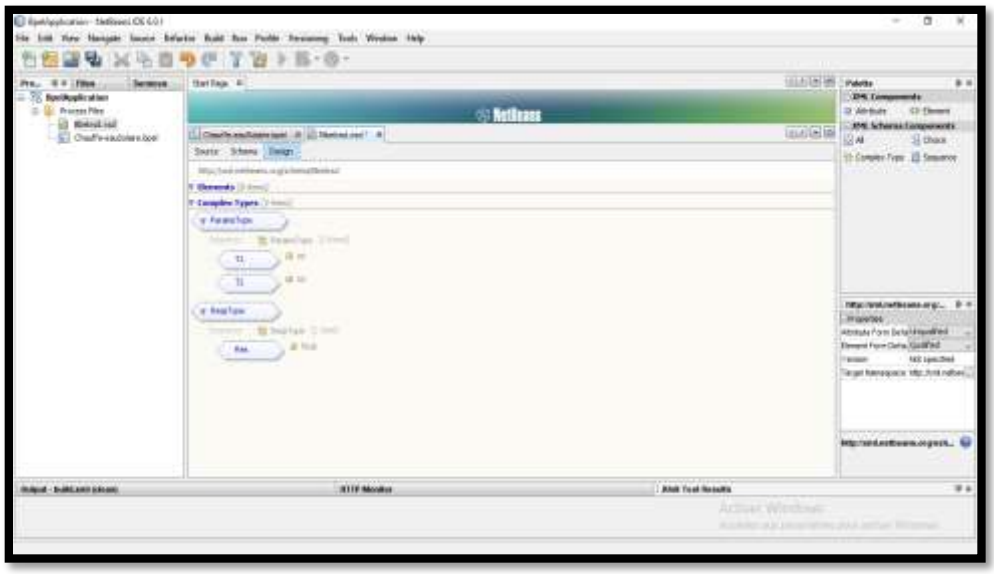
Ou bien : click droit dans l'endroit :***ComplexTypes→*Add→*ComplexType*** :

- Définir deux types complexes : ***ParamsType*** et ***RespType***.
- ***ParamsType***avecdeux éléments ***entiers*** nommés (***T1, T2***).
- ***RespType***avec un élément***float*** nommé (***Res***).

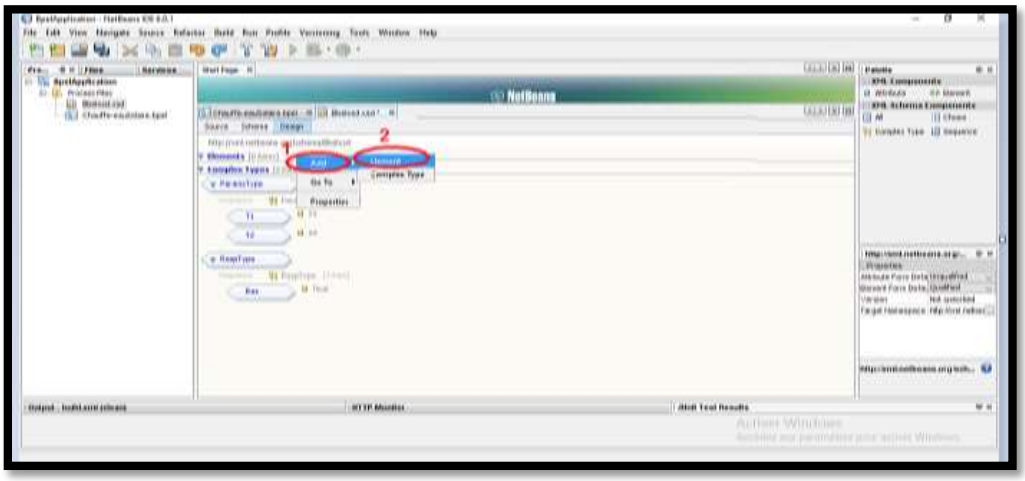
- Puis de la même façon créer ***Resp***, suivant même étapes que ***T1, T2*** pour créer ***Res***.

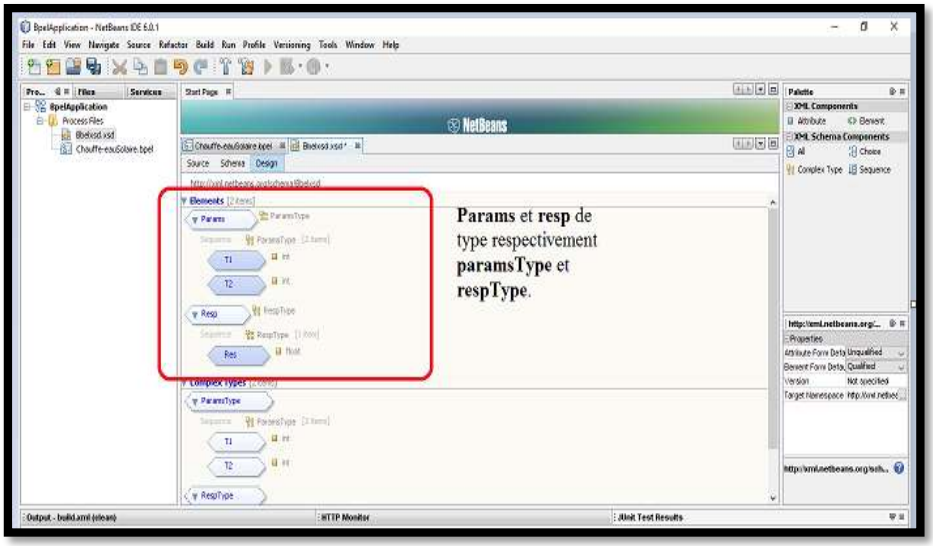


Pour défini les types des valeur

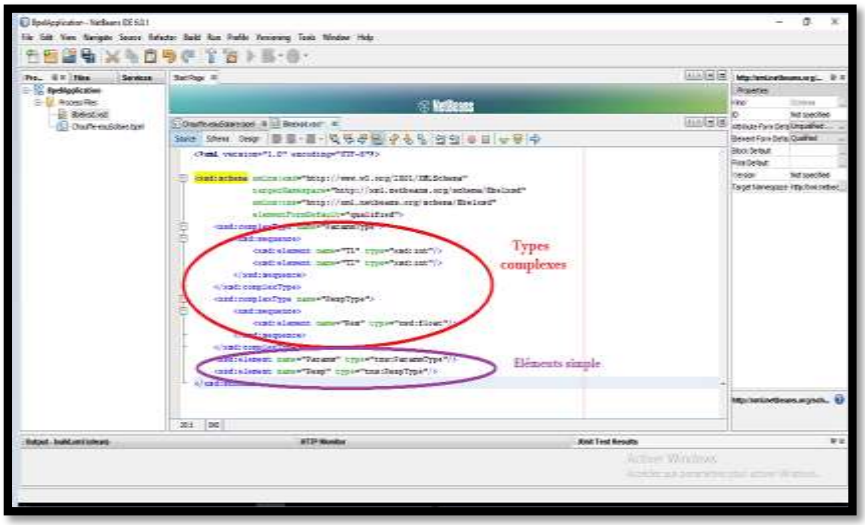


- Créer ensuite deux éléments simple :***Params*** et ***Resp*** de type respectivement ***ParamsType***et***RespType***, comme suit :





Le fichier ***XSD*** final :



5.3 **Création de services partenaires :**

Ce processus ***BPEL*** interagit avec deux services.

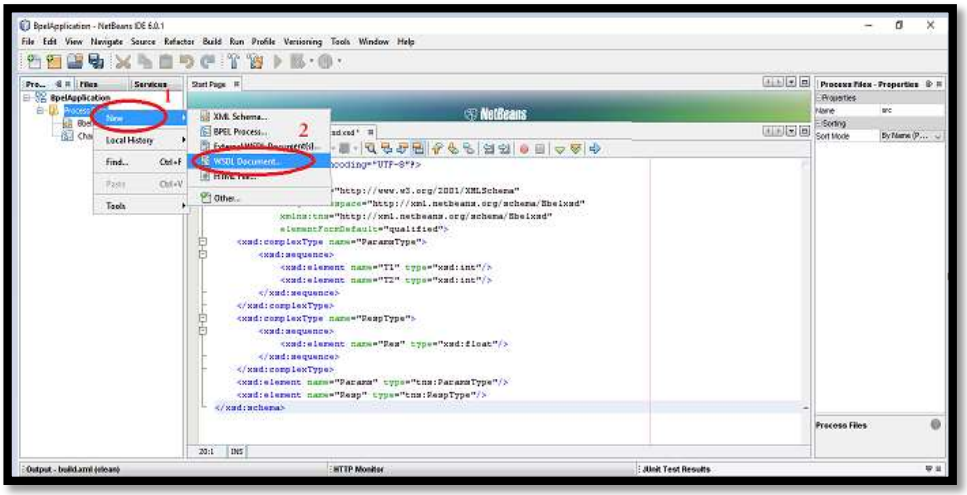
1- Un ***service client*** qui va fournir les entrées au processus sera décrit par un fichier ***WSDL***(Web Services Description Language).

2- Un ***service externe*** qui réalise le calcule.

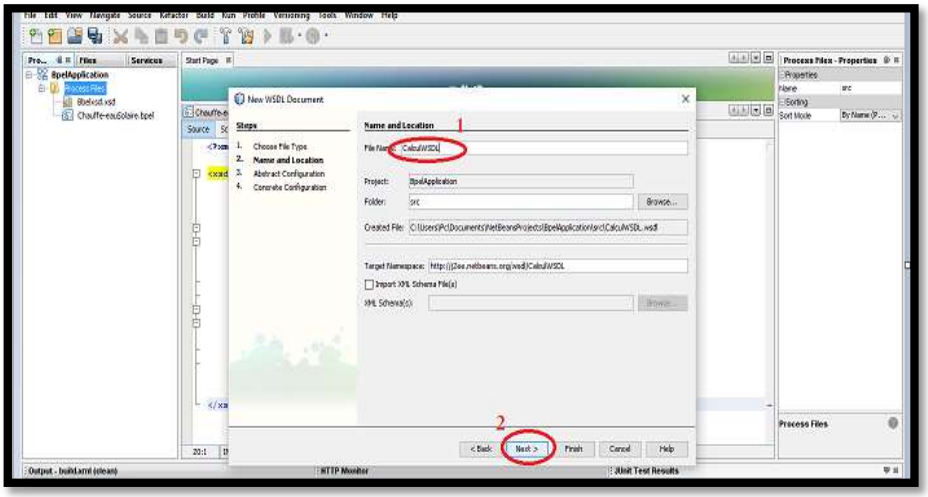
Création de service client :

Ce service est décrit par le fichier ***WSDL***, pour le créer :

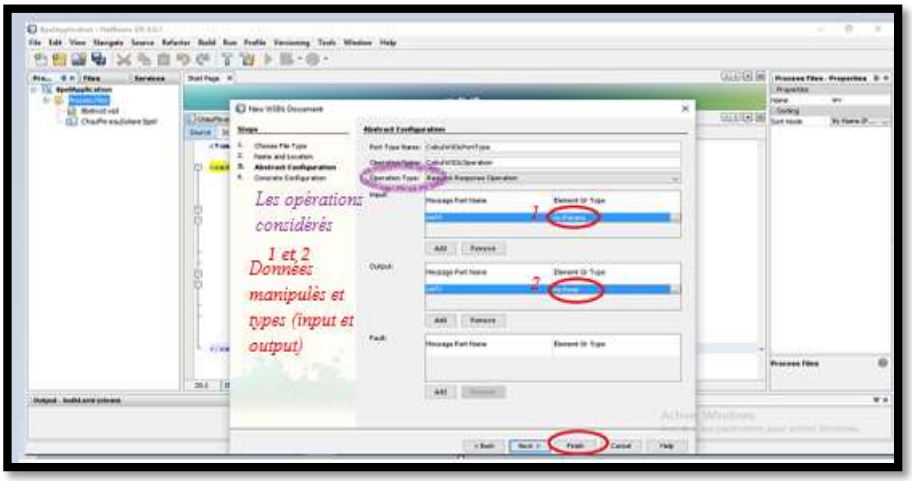
- Faire un clic droit sur ***Process Files*** *et choisir* ***New→WSDLDocument***



❖ Appeler ce fichier ***calculWSDL*** **:**



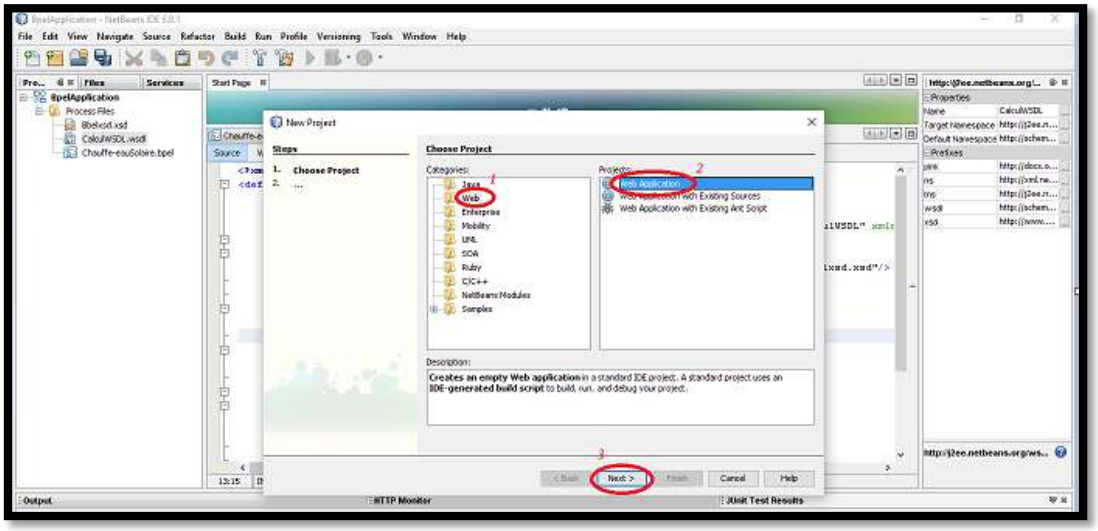
❖ Dans la partie ***Abstract Configuration***, définir le type des ***entrées*** et des ***sorties*** respectivement dans les cases ***Input*** et ***Output*** :



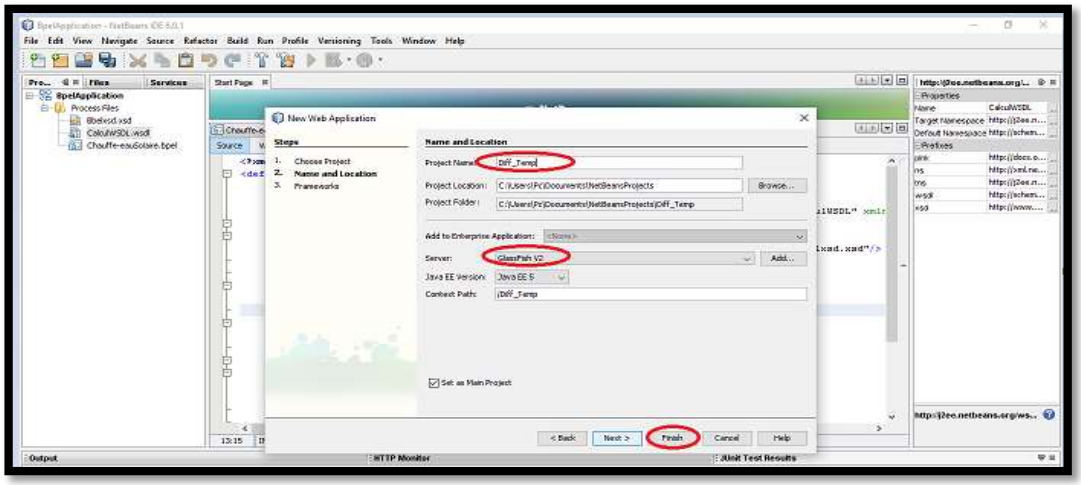
Création de service web externe :

Créer un projet web :

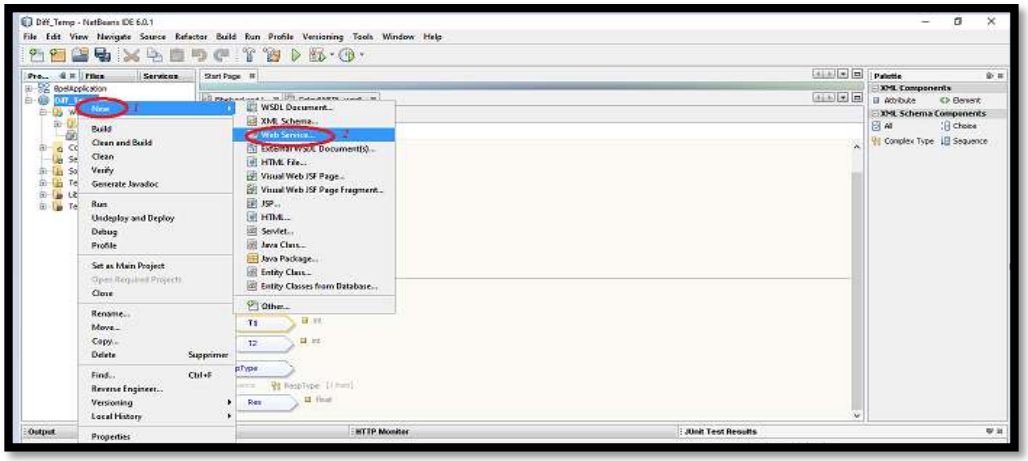
- **New → Java web →Web Application :**



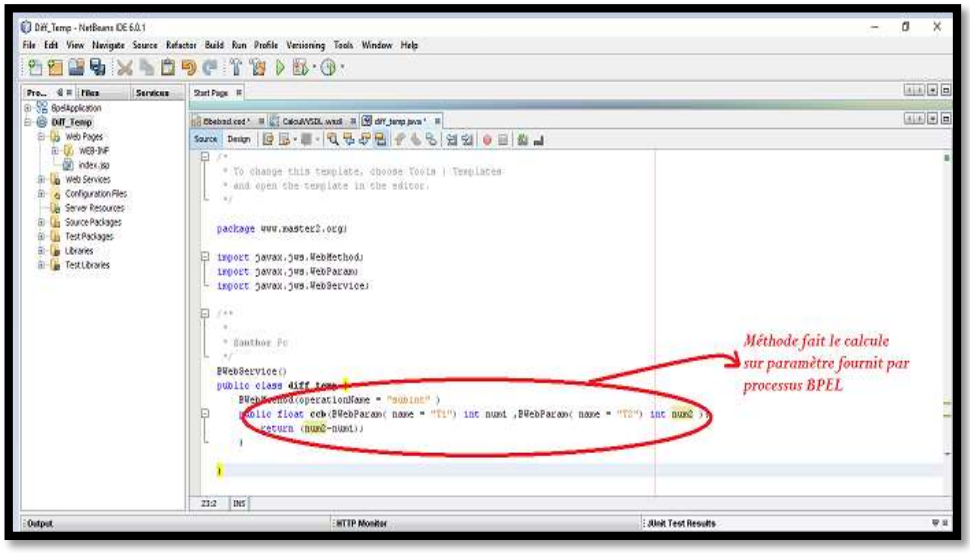
Choisir ***Diff_Temp***comme nom ***:***



- Créer ensuite un nouveau ***service web*** intitulé ***Diff_Temp*** :



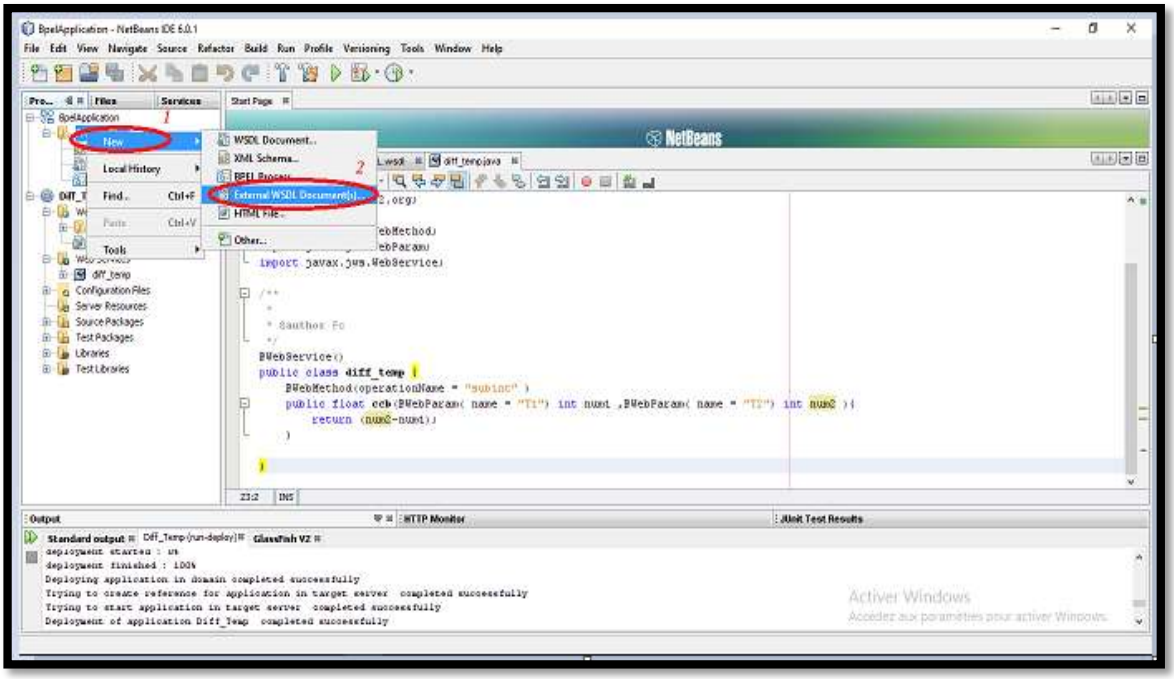
- Ce service contient une seule méthode, ***Subint***, qui récupère deux entiers, et retourne le résultat de calcule cette opération est représenté par une méthode java.
- **Enregistrer et Déployer ce service.**



5.4 **Préparation du processus BPEL :**

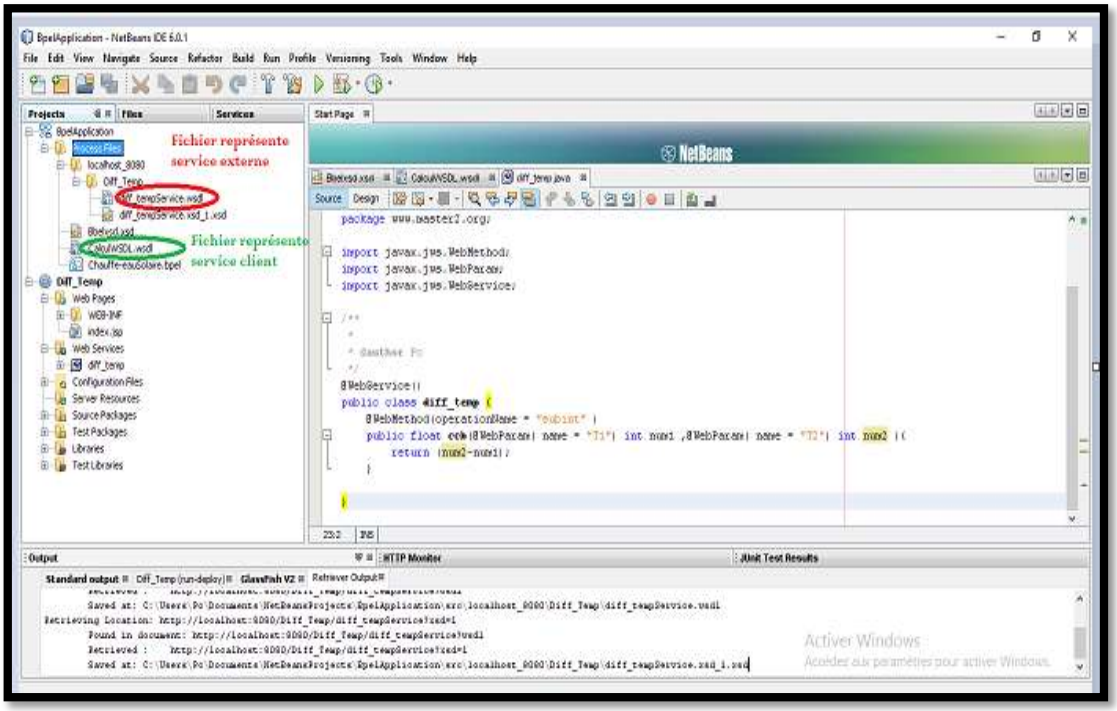
- Pour représenter un nouveau service web dans le processus ***BPEL***, il faut le représenter sousforme de fichier ***WSDL*** dans le projet ***BPEL***.
- Faire un clic-droit sur ***Process Files***del'application***BpelApplication***, *et* choisir

  New→ExternalWSDLDocument(s)

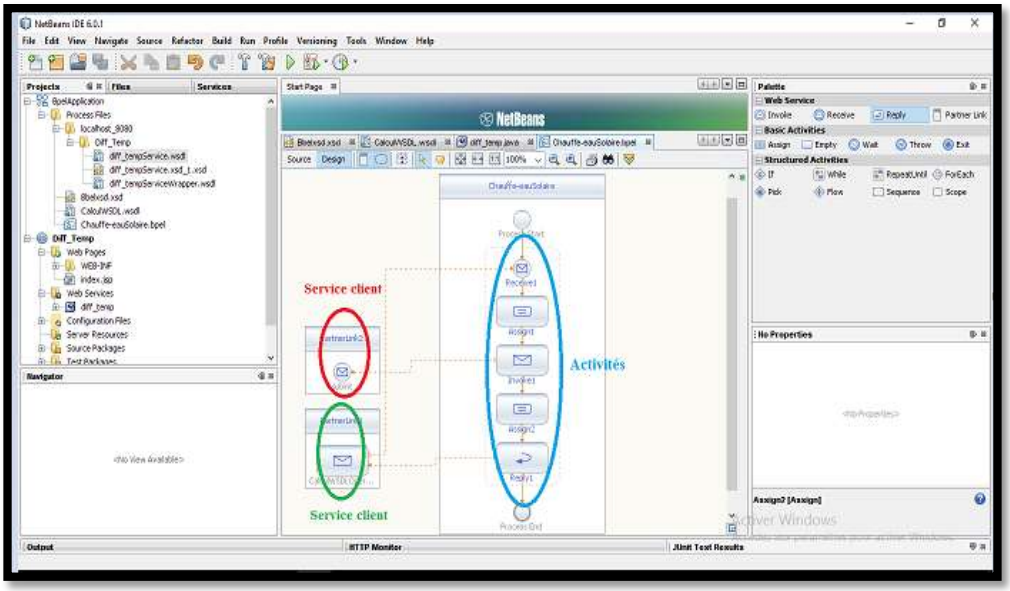


- Lui donner comme ***URL*** le chemin vers le fichier ***WSDL*** de votre service web :

  http://localhost:8080/Diff_Temp/diff_tempService?wsdl, et résultat comme suite :



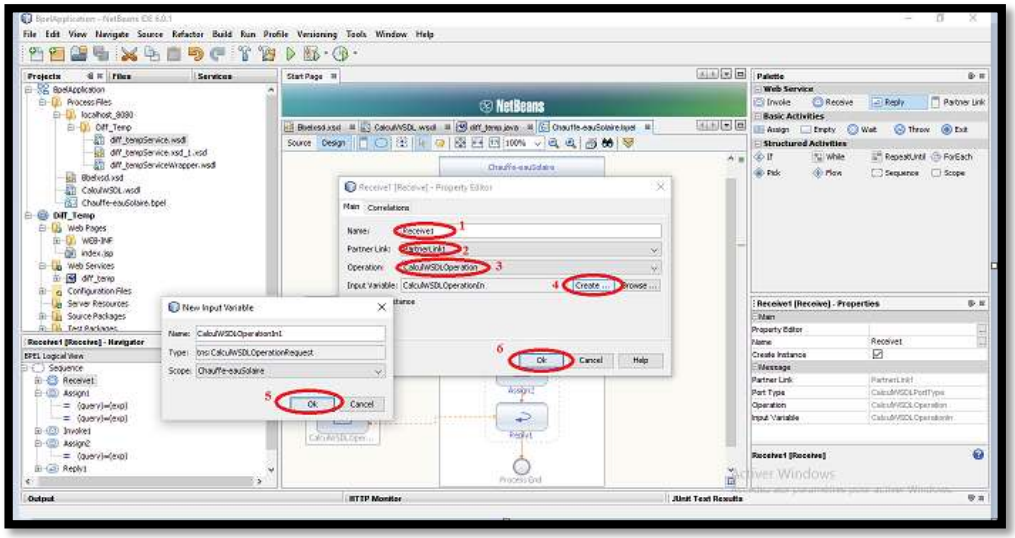
- Ouvrir le fichier : ***Chauffe-eauSolaire.Bpel***
- Faire glisser ensuite votre fichier ***WSDL*** importé vers la fenêtre principale.
- Faire glisser le fichier ***WSDL*** du service client vers la fenêtre principale (service qui fournit les entrées).
- Faire glisser ensuite les activités trouver dans la palette à droite :***Receive***,***Assign1***, ***Assign2***, ***Invoke***, ***Reply*** entre les activités ***Processstart*** et ***Processend***.
- Relie le processus ***Bpel*** avec les deux services.
- Le fichier apparait comme ceci :



Puis configurer chaque activité.

- **Receive1**: pour récupérer les trois paramètres en entrée à partir du service décrit par ***CalculWSDL***.

  - Double clic sur cette activité pour la configurer.



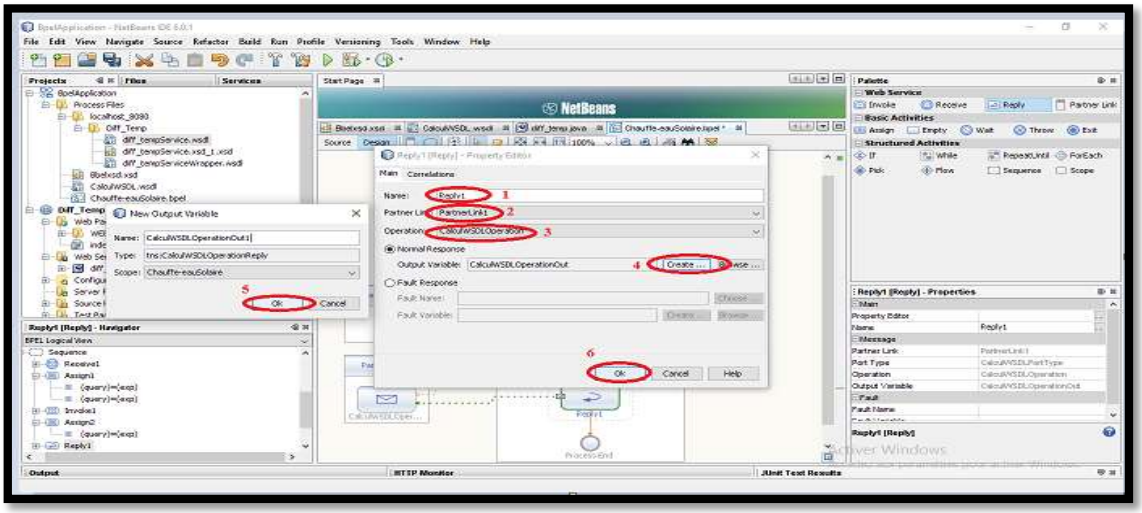
  - Préciser le ***Partner Link*** à partir duquel les données seront reçues (***PartnerLink1***),

  Partenerlink représente le service.

  - l'opération visée (***CalculWSDLOperation***) et générer la variable en entrée en cliquant sur le bouton ***Create***.

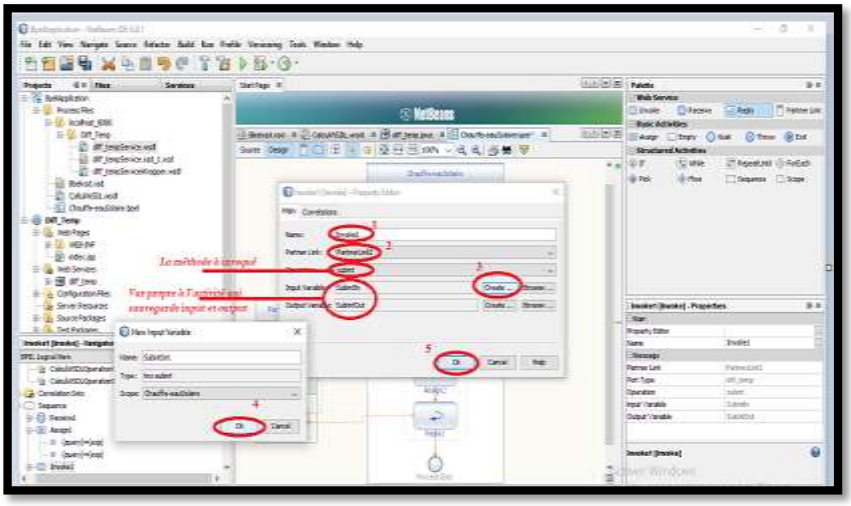
- La fenêtre aura l'apparence suivante :

- **Reply1 :** pour envoyer le résultat du processus au ***service client***. La configurer de la même manière que l'activité ***Receive1***.

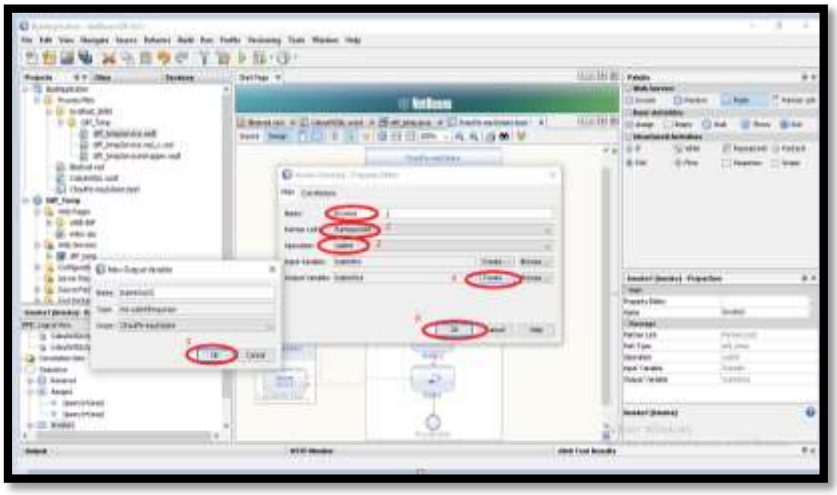


- ***Invoke1*** **:**cetteactivitépermet de faire appel à un ***service externe***.

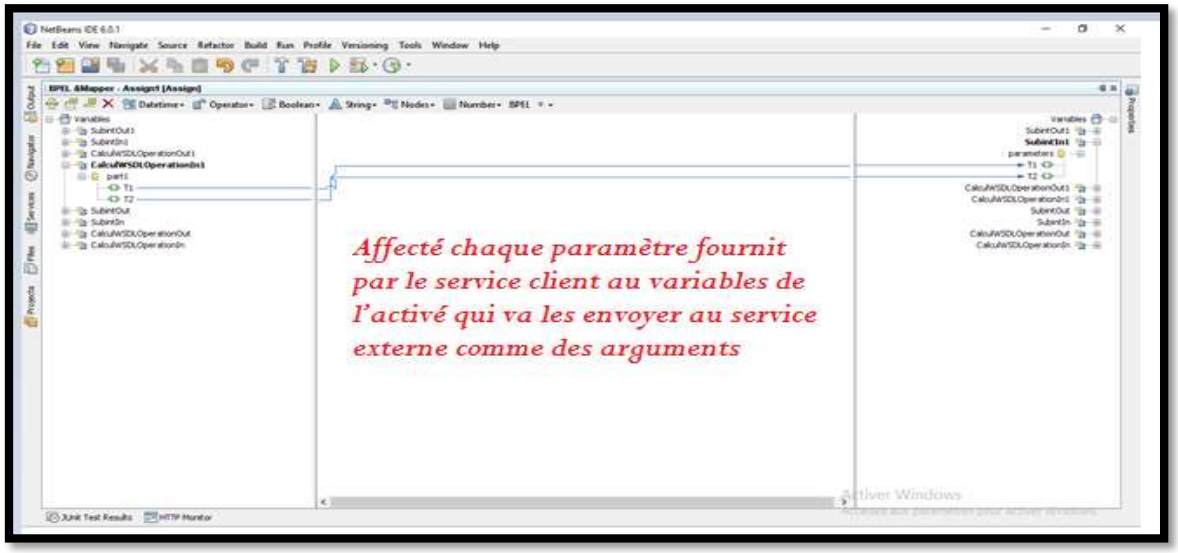
- Configurer l'activité ***Invoke1*** comme indiqué dans la figure suivante :



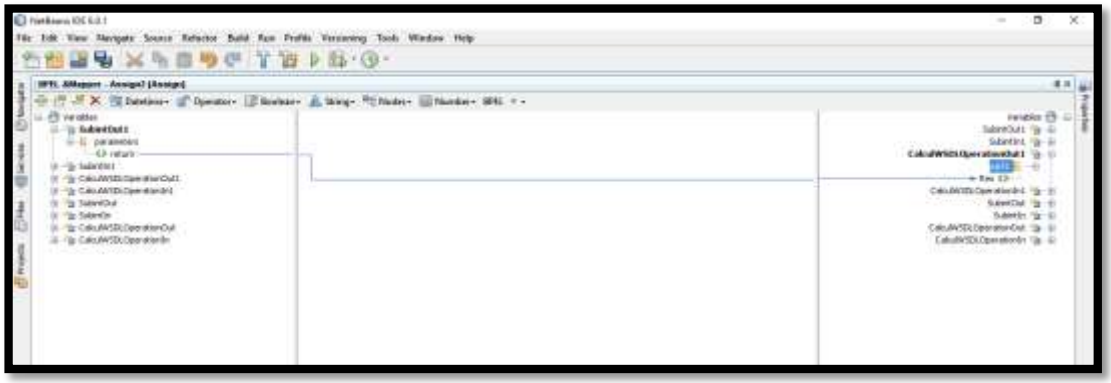
Même façon de création deuxième :



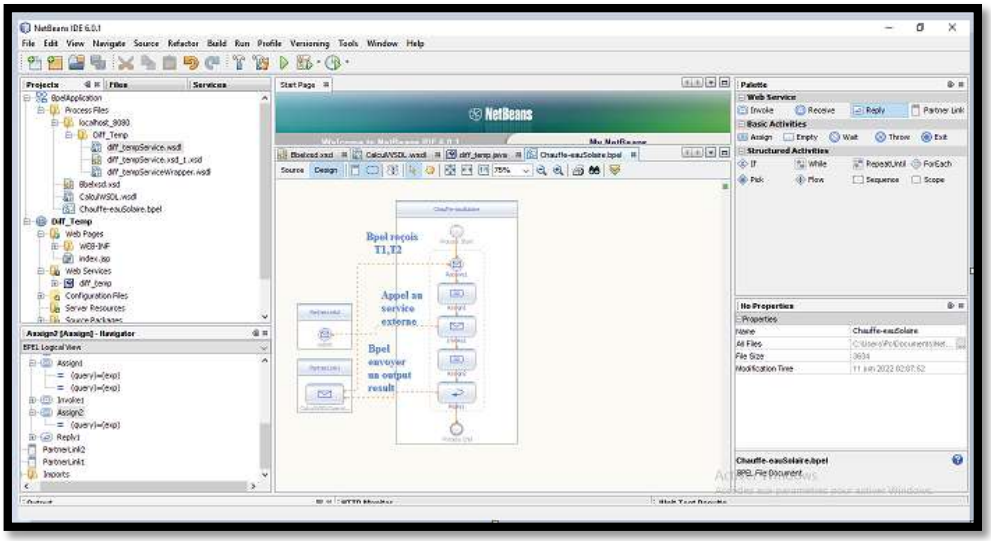
- Insérer ensuite deux activités ***Assign*** **(*Assign1*** et ***Assign2*****)** respectivement entre ***Receive1*** et ***Invoke1***, et entre ***Invoke1*** et ***Reply1***.
- L'activité ***Assign***permet d'affecter les variables en ***entrée*** aux variables en ***sortie*** du processus.
- Configurer le premier ***Assign1*** *comme suit :*



- Configurer le deuxième ***Assign2*** *:*

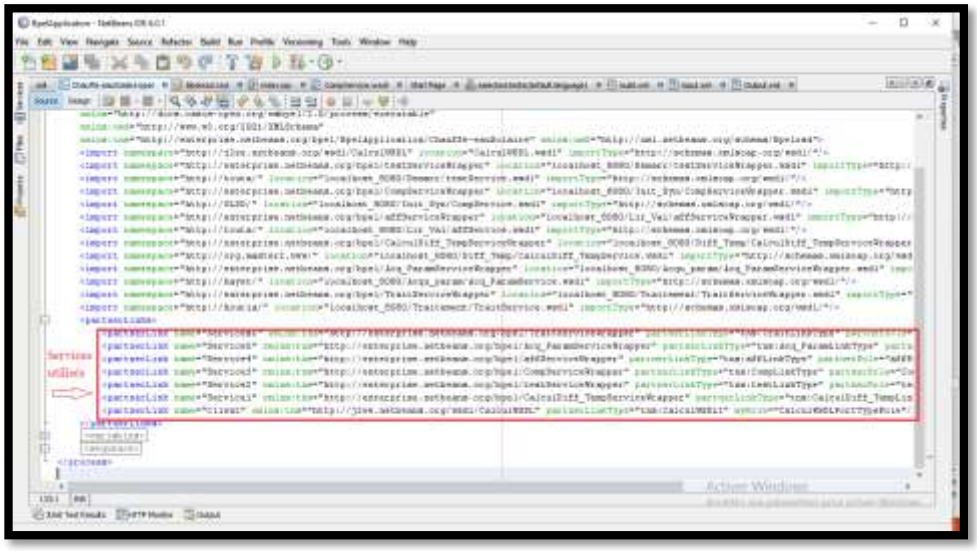


- **Le processus obtenu aura alors l'allure suivante :**

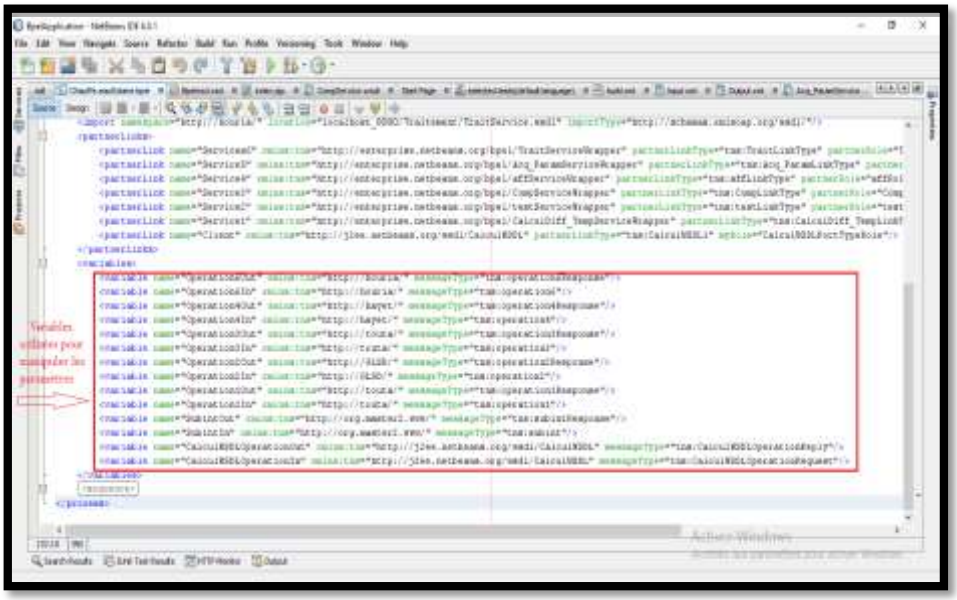


- **BPEL source :**

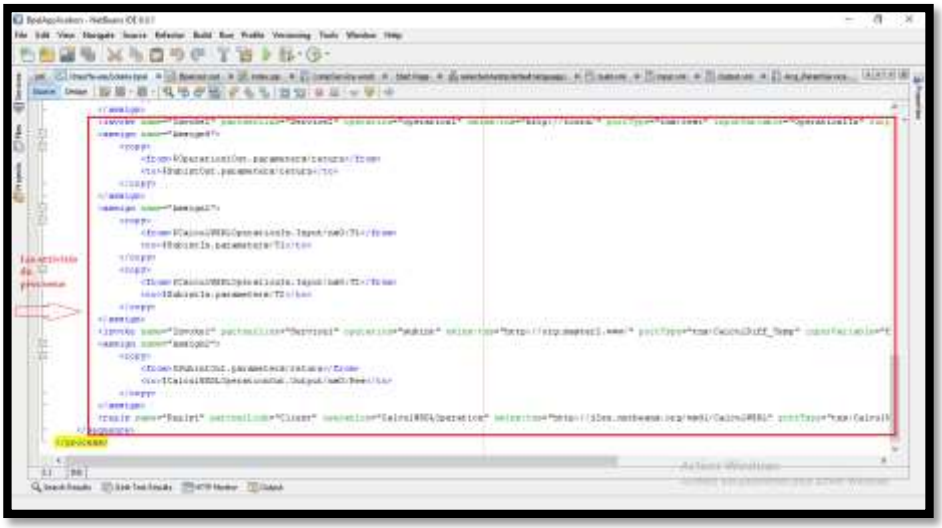
Les services utilisés :



Variables utilisées pour manipuler les paramètres :



Les activités du processus :



5.5 Création de l'Application Composite :

Pour créer l'application composite :

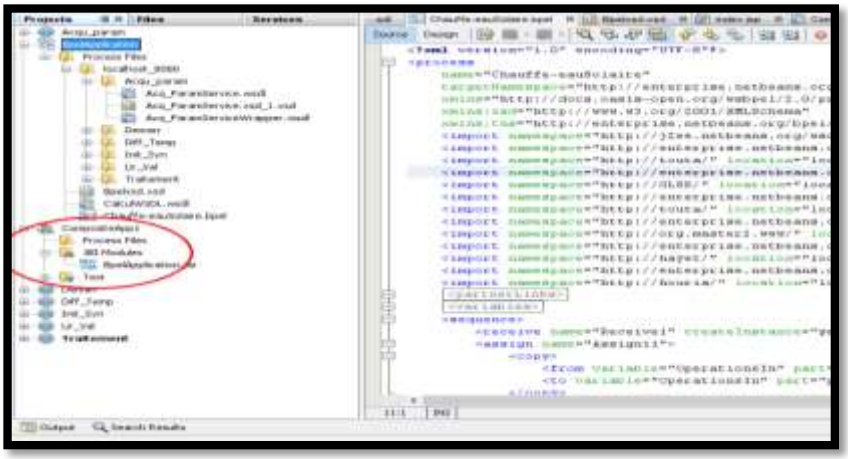
-Choisir :***NewProject →SOA →CompositeApplication***

- La nommer ***CompositeApp1***.
- La fenêtre qui apparaitre est divisée en 3 parties : une pour les ports ***WSDL***, une pour les ***modules JBI,*** et une autre pour les ***modules externes***.

• Faire glisser votre application ***BpelApplication***dans la partie JBI Modules.

ou bien : clic droit ***→Add JBI modul →***sélectionnez votre***projet***etcliquez sur *:*

*« **Addproject JAR files** ».*



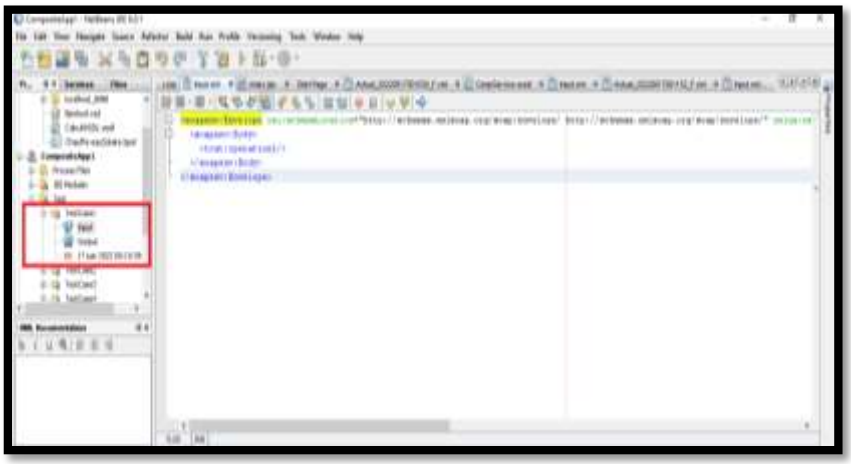
• ***Enregistrer*** et ***Déployer*** votre application.

5.6 Test de l'application composite :

• Une fois votre application ***déployée***, il possible de la ***tester***.

- Cliquer avec le bouton droit sur le répertoire ***Test*** *de* ***l'application composite****, et* choisir : ***New Test Case*** *(L'appeler:TestCase1).*

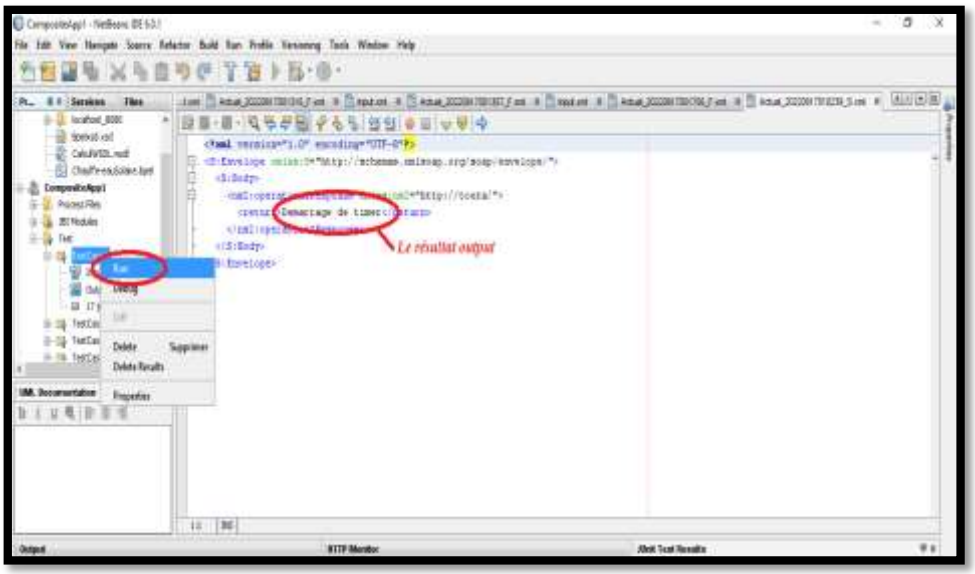
- Dans la fenêtre suivante, choisir le fichier ***WSDL*** de votre application composite, puis l'opération *Operation1*.



- Clic-droit sur ***TestCase1*** *et lancer le* ***test*** *:*

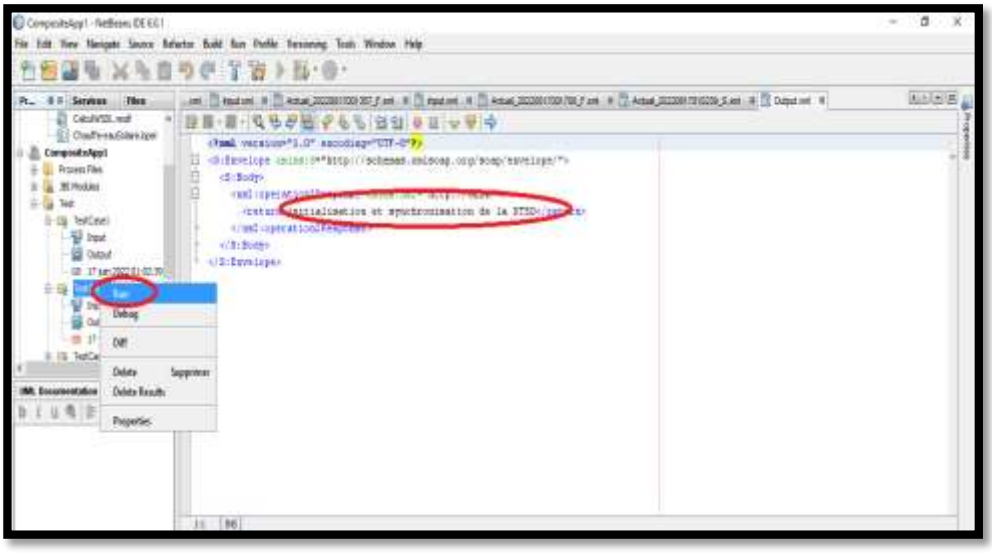
- le fichier Output sera contient le résultat.

(La première fois que vous exécutez votre test, les résultats signalent correctement l'échec. Si vous exécutez à nouveau le test sans modifier l'entrée, la deuxième et les exécutions suivantes signalent le succès).

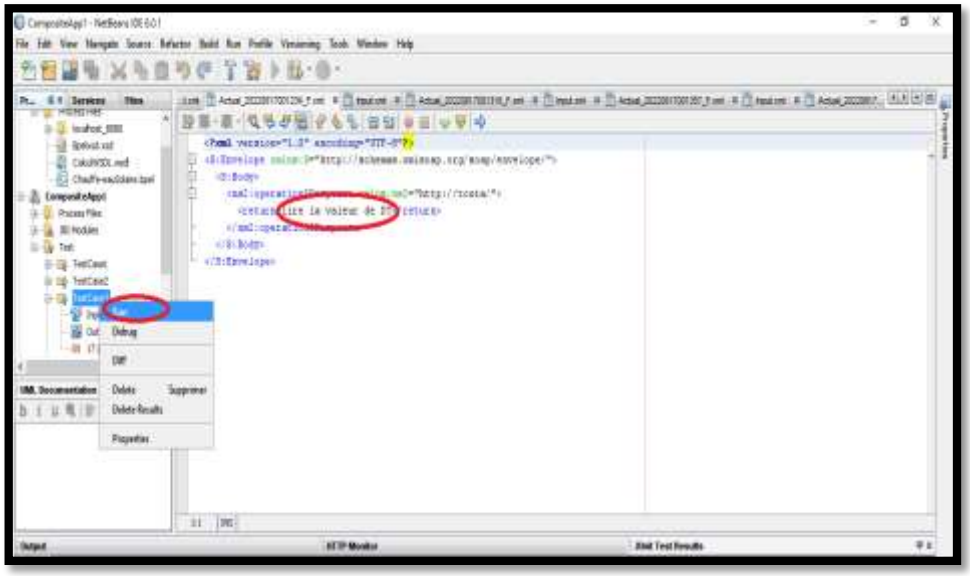


Les restes services de la même façon de teste :

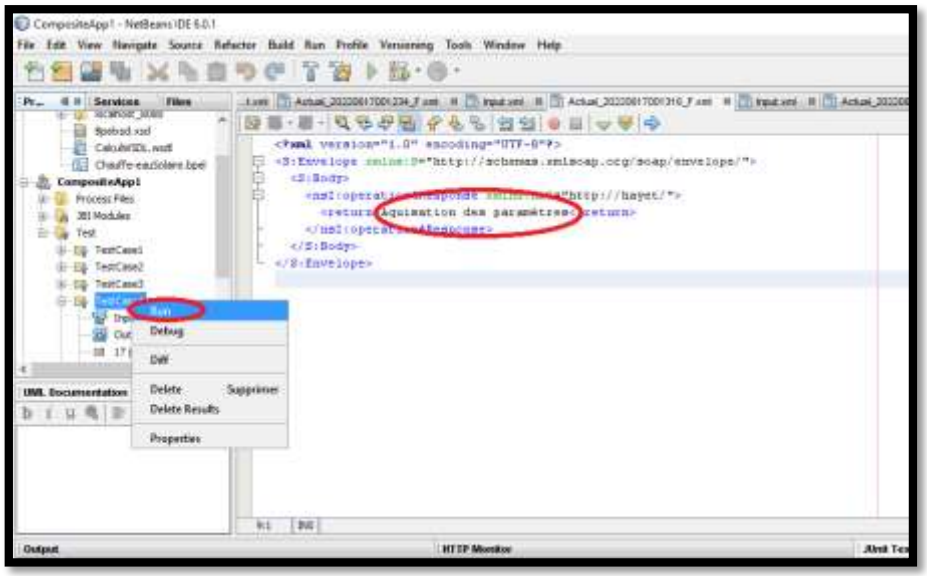
- Clic-droit sur ***TestCase2*** *et lancer le* ***test*** *:*



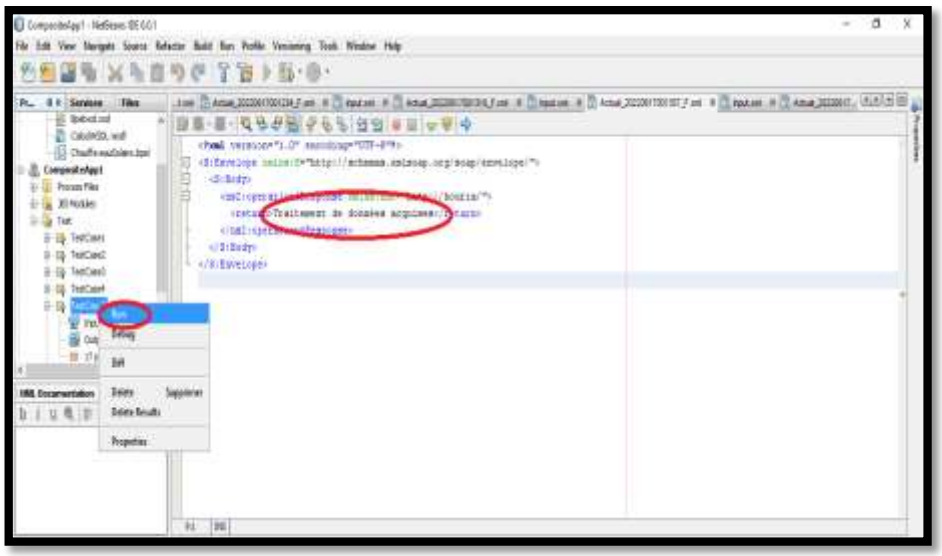
- Clic-droit sur ***TestCase3*** *et lancer le* ***test*** *:*



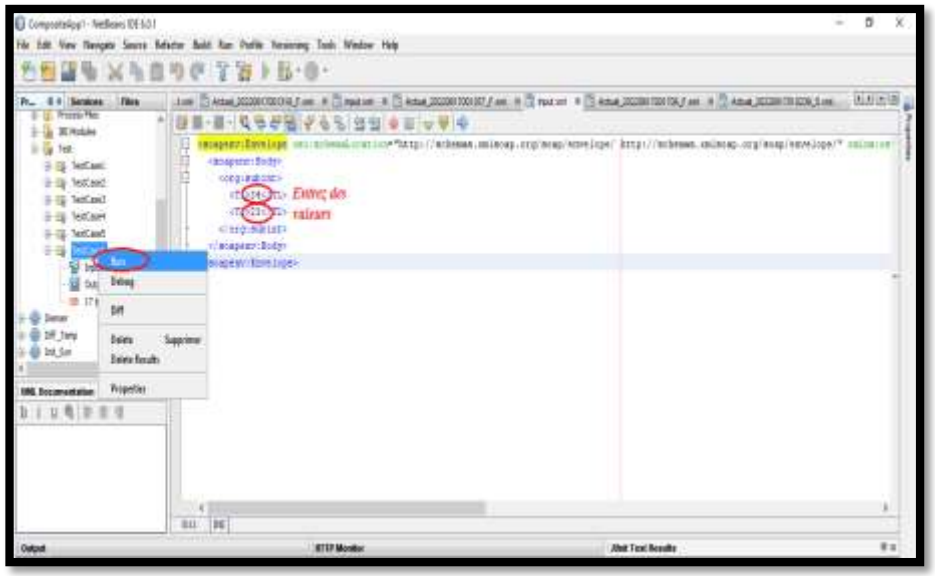
- Clic-droit sur ***TestCase4****et lancer le* ***test*** *:*



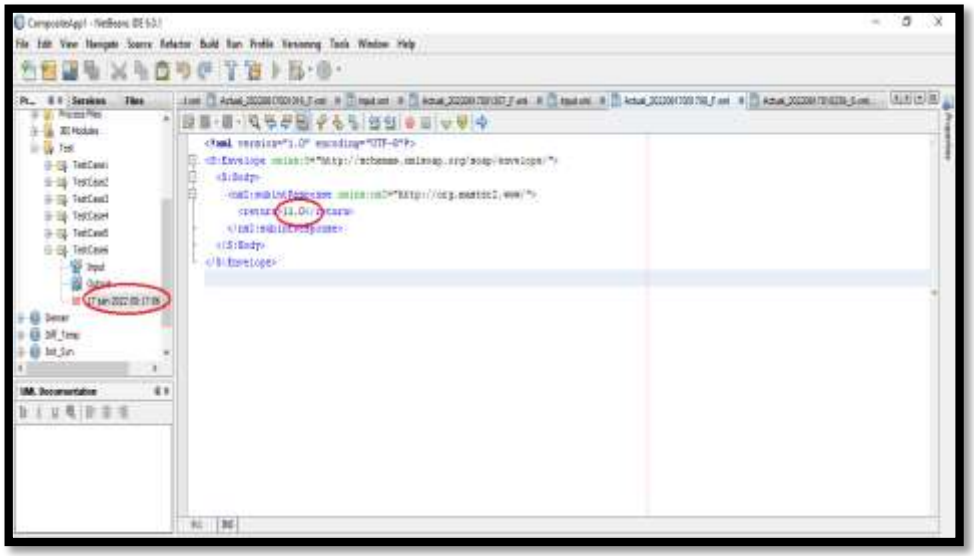
- Clic-droit sur ***TestCase5*** *et lancer le* ***test*** *:*



- Clic-droit sur ***TestCase6*** *et lancer le* ***test*** *:*



- Donc :le fichier ***Output*** sera contient le résultat comme suit :



Code source ***HTML*** :

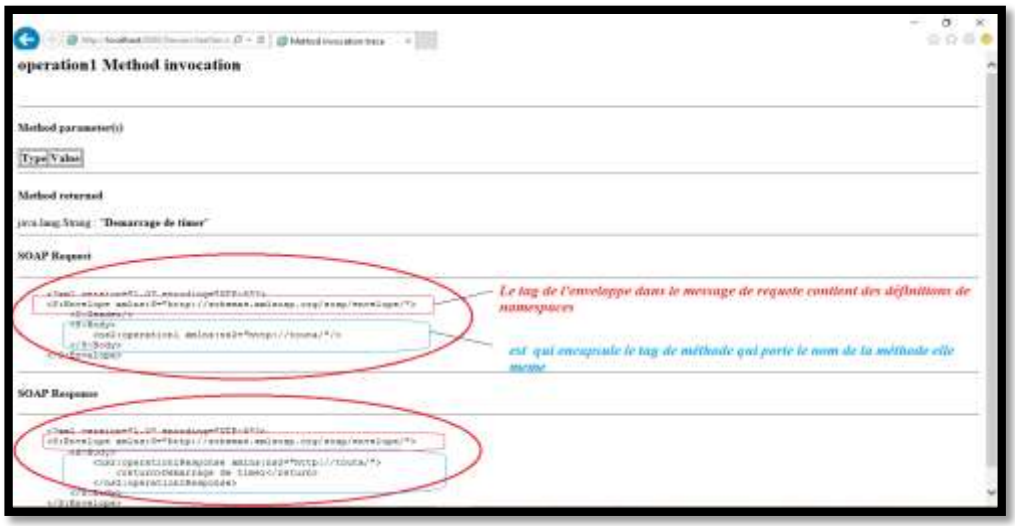
Résultat sur web :

Double clic sur le nom de service «***Demarrage de Timer*** » pour avoir les messages **de l'operation1** :

Opération1_Méthode invocation :

- Dans la requête en bas, on retrouve bien les deux éléments obligatoires caractéristiques d'un message SOAP. Le tag de l'enveloppe dans le message de requête contient des définitions de namespaces. On trouve ensuite le tag ***S. Body*** qui encapsule le tag de méthode qui porte le

nom de la méthode elle-même (ou le même nom suivi de "***Response***" dans le message de réponse.



Même façon avec le reste services :

Opération2 _Méthode invocation :

- Double clic sur le nom de service «***initialisation et synchronisation de la DT50*** » pour avoir les messages **de l'operation2** :

Opération3_ Méthode invocation :

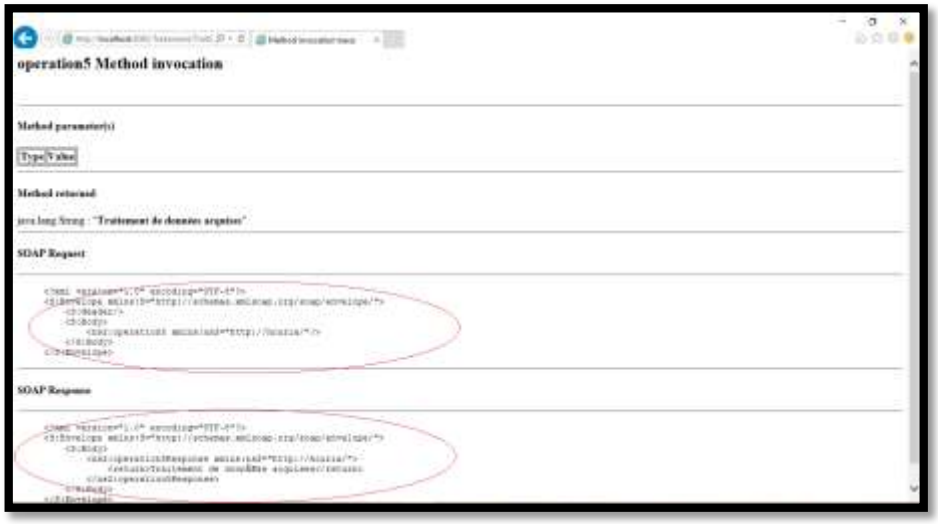
- Double clic sur le nom de service «***lire la valeur de DT***» pour avoir les messages **de l'operation3** :

Opération4_ Méthode invocation :

- Double clic sur le nom de service «***Acquisition des paramètres***» pour avoir les messages **de l'operation4** :

Opération5_ Méthode invocation :

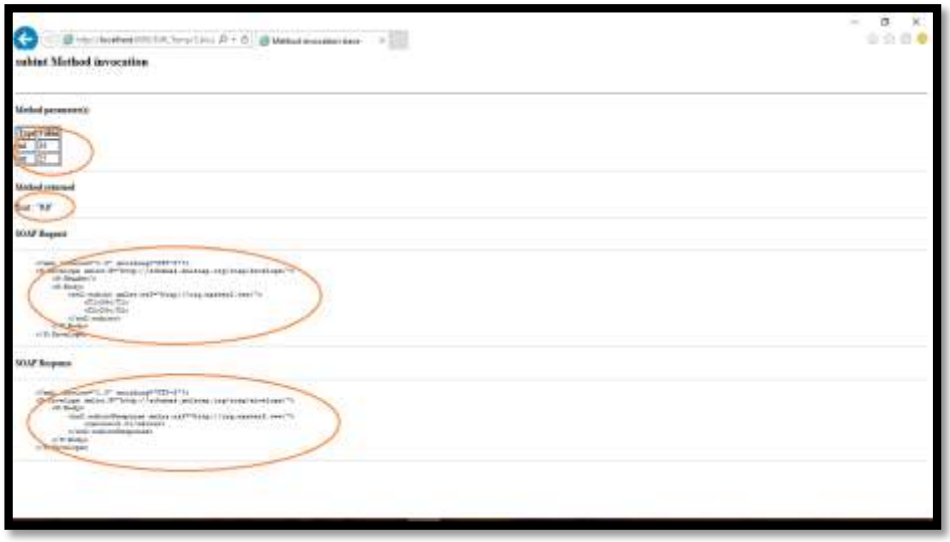
- Double clic sur le nom de service «***Traitement de données acquises***» pour avoir les messages **de l'operation5** :



Double clic sur le nom de service «Différence de Température» pour avoir les messages **de Opération-subint_ Méthode invocation :**

- **l'operation-subint** : saisi des valeurs

L'Opération -subint_ Méthode invocation est comme suite :



6. Conclusion :

Nous avons utilisé la technologie des services web pour structure notre cas d'étude en un ensemble de services web qui exposent leur interface fonctionnelle et qui communiquent par messages. Nous avons profité de sa possibilité de développer des applications par composition de services élémentaires en utilisant le workflow.

Nous avons essayé de donner une vision globale sur la réalisation de notre application et cela par la présentation de quelques interfaces principales dont le rôle répond à un besoin parmi ceux qui ont été décrit à la phase de conception.

Références Bibliographiques :

[1] By Heather Kreger, IBM Software Group, May 2001.p:6.

[2] W3C, ServicesWeb.-2004.-http://www.w3.org/TR/2004/NOTE-ws-arch-20040211

[3] https://openclassrooms.com/courses/lesservicesweb 3/21.

[4] Biplav Srivastava and Jana Koehler. Web service composition - current solutions and open problems. In ICAPS 2003 Workshop on Planning for Web Services, pages 28-35, 2003.

[5] Stephan Hagemann, Carolin Letz, and Gottfried Vossen. Web service discovery : Reality check 2.0. Next Generation Web Services Practices, International Conference on, pages 113 118, 2007.

[6] Helga Duarte, Marie-Christine Fauvet, Marion Dumas, and BoualemBenatallah. Vers un modèle de composition de services web avec propriétés transactionnelles. Ingénierie des Systèmes d'Information, 10(3) :9-28,2005.

[7] Michael Mrissa. Médiation Sémantique Orientée Contexte pour la Composition de Services Web. PhDthesis, Université Claude Bernard Lyon l, November 2007.

[8] NereaArenaza. Composition semi-automatique de services web. Projet de mastèr, École polytechnique fédérale de Lausanne, 2006.

[9] Amel Benna, NacerBoudjlida, and HassinaTalantikite. SAWSDL, mediation and XQueryfror web services discovery. In NOTERE '08:Proceedings of the 8th international conferenceon new technologies in distributedsystems, pages 1-10, New York, NY, USA, 2008. ACM.

[10] Alexandre Alves, AssafArkin, Sid Askary, Charlton Barreto, Ben Bloch, Francisco Curbera, Mark Ford, YaronGoland, Alejandro Gufzar, NeelakantanKartha, Canyang Kevin Liu, Rania Khalaf, Dieter K6nig, Mike Marin, VinkeshMehta, SatishThatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu. Web services business process executionlanguage version 2.0. Technical report, OASIS, 2007.

[11] Thomas Erl; John Evdemon; Diane Jordan; KhanderaoKand; Dieter K6nig; Simon Moser; Ralph Stout; Ron Ten-Hove; Ivana Trickovic; Danny van der Rijn et Alex Yiu Charlton Barreto;, VaughnBullard;. Web services business processexecutionlanguage version 2.0 primer. Technical report, OASIS, 2007.

[12] Benny Mathew et PoornachandraSarangMatjaz B. Juric. Business ProcessExecutionLanguage for Web Services. PACKT, second edition, Janvier 2006.

[13]http://www.wfmc.org/.

[14] James Michaelis, Li Ding, and Deborah McGuinness. Towards the Explanation of workflow. In Proceedings of the IJCAI'09 Workshop on Explanation- AwareComputing., 2009.

[15] Wil Van Der Aalst and Kees M. van Hee. Workflow Management : Models, Methods, and Systems. MIT Press, 2002.

[16] Ronni Marshak. Requirements for workflow products. In David D. Coleman and Morgan Kaufman, editors, Proceedings of Groupware 92. August 1992.

[17] Wil Van Der Aalst, Arthur ter Hofstede, BartekKiepuszewski, and Alistair Barros. Workflow patterns. Distributed and ParallelDatabases, 14(1) :5–51, 2003.

[18] Clarence Ellis, Simon Gibbs, and Gail Rein. Groupware : Some issues and experiences. In Communications in the ACM, pages 38–58, 1991.

[19] Anna Kalenkova, Massimiliano de Leoni, and Wil M. P. van der Aalst. Discovering, analyzing and enhancing BPMN modelsusingprom. In Proceedings of the BPM Demo Sessions 2014 Co-locatedwith the 12th International Conference on Business Process Management (BPM 2014), Eindhoven, The Netherlands, September 10, 2014., page 36, 2014.

[20] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Process and deviation exploration with inductive visualminer. In Proceedings of the BPM Demo Sessions 2014 Co-locatedwith the 12th International Conferenceon Business Process Management (BPM 2014), Eindhoven, The Netherlands, September 10, 2014., page 46, 2014.

[21] EwaDeelman, Dennis Gannon, Matthew S. Shields, and Ian Taylor. Workflows and e-science : An overview of workflow system features and capabilities. Future GenerationComp. Syst., 25(5) :528–540, 2009.

[22] GideonJuve. Resource Management for ScientificWorkflows. Phdthesis, University of SouthernCalifornia, May 2012.

[23] Rafael Ferreira da Silva, Weiwei Chen, GideonJuve, KaranVahi, , and EwaDeelman. Communityresources for enablingresearch in distributedscientificworkflows. In 10th IEEE International Conference on e-Science, 2014.

[24] JiaYu and RajkumarBuyya. A taxonomy of workflow management systems for gridcomputing. Technical report, Journal of gridComputing, 2005.

[25] Wil Van Der Aalst, Arthur ter Hofstede, BartekKiepuszewski, and Alistair Barros. Workflow patterns. Distributed and ParallelDatabases, 14(1) :5–51, 2003.

[26] Ketan Maheshwari. Data-intensive ScientificWorkflows : Representation of Parallelism and Enactments on DistributedSystems. PhDthesis, Ecole Polytech Universitaire, Sophia Antipolis, Nice, 2011.

[27] JiaYu. QoS-basedScheduling of Workflows on Global Grids. Ph.d. thesis, 2007.

[28] http://www.w3.org/XML/

[29] Gregor Von Laszewski, Kaizar Amin, MihaelHategan, Nestor J. Zaluzec Shawn Hampton, and Albert Rossi. Gridant : A client-controllablegrid workflow system. In in 37th HawaiÕi International Conference on System Science, Island of Hawaii, Big Island, pages 5–8, 2004.
[30] Frank Leymann. Web services flow language (wsfl 1.0). Technical report, IBM, 2001. Availableat : http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[31] Tony Andrews et al. Business Process Execution Language for Web Services, Version 1.1. Technical report, BEA Systems, IBM, Microsoft, May 2003.

[32] http://www.w3.org/TR/xml-pipeline/

[33] RajkumarBuyya and Srikumar Venugopal. The gridbustoolkit for service orientedgrid and utility computing : An overview and status report. In Euro-Par 2007 ParallelProcessing. Press, 2004.

[34] https://research.cs.wisc.edu/htcondor/dagman/dagman.html.

[35] http://www.informatik.uni-hamburg.de/TGI/PetriNets/

[36] http://www.uml.org/

[37] Marco Montali, Fabrizio Maria Maggi, Federico Chesani, Paola Mello, and Wil M. P. van der Aalst. Monitoring business constraintswith the eventcalculus. ACM TIST, 5(1) :17, 2013.

[38] Ralph Mietzner, Christoph Fehling, DimkaKarastoyanova, and Frank Ley-mann. Combining Horizontal and Vertical Composition of Services. In Pro-ceedings of IEEE International Conference on Service-OrientedComputing and Applications (SOCA 2010). IEEE, December 2010.

[39] Dario Bruneo, Salvatore Distefano, Francesco Longo, and Marco Scarpa.Stochasticevaluation of qos in service-basedsystems. IEEE Trans. ParallelDistrib. Syst., 24(10) :2090–2099, 2013.

[40] GideonJuve and EwaDeelman. Resource provisioning options for large-scalescientificworkflows. In 3rd International Workshop on ScientificWorkflows and Business Workflow Standards in e-Science (SWBES 08), 2008.

[41]Florian Lautenbacher and Bernhard Bauer. A survey on workflow annota-tion& composition approaches. In Martin Hepp, Knut Hinkelmann, DimitrisKaragiannis, Rüdiger Klein, and Nenad Stojanovic, editors, Proceedings of the Workshop on Semantic Business Process and Product LifecycleManage-ment SBPM 2007, held in conjunctionwith the 3rd EuropeanSemantic Web Conference (ESWC 2007), Innsbruck, Austria, June 7, 2007, volume 251 ofCEUR Workshop Proceedings. CEUR-WS.org, 2007.

[42] Hanna Eberle, Frank Leymann, and Tobias Unger. ImplementationArchi-tectures for Adaptive Workflow Management. In ADAPTIVE 2010, pages 1–6. XpertPublishing Services, November 2010.

[43] Jan Hidders, PaoloMissier, and JacekSroka, editors. FundamentaInformati-cae – Special issue on Scalable Workflow Enactment Engines and Technology , volume 128. IOS Press, 2013.

[44] MatjazJuric, Benny Mathew, and PoornachandraSarang. Business Process Execution Language for Web Services 2nd Edition. PacktPublishing, 2006.